Simulation Relations among Message Passing and Mobile Agent Algorithms

Maria Ioanna Spyrakou
A0013

Examination committee:
Aris Pagourtzis, School of Electrical and Computer Engineering, National Technical University of Athens.
Dimitris Fotakis, School of Electrical and Computer Engineering, National Technical University of Athens.
Stathis Zachos, School of Electrical and Computer Engineering, National Technical University of Athens.

Supervisor:
Aris Pagourtzis, Professor,
School of Electrical and Computer Engineering ,
National Technical University of Athens.

Η παρούσα Διπλωματική Εργασία
εκπονήθηκε στα πλαίσια των σπουδών
για την απόκτηση του
Μεταπτυχιακού Διπλώματος Ειδίκευσης
«Αλγόριθμοι, Λογική και Διακριτά Μαθηματικά»
που απονέμει το
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
του
Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών

Εγκρίθηκε την ..... από Εξεταστική Επιτροπή
αποτελούμενη από τους:

Ονοματεπώνυμο	**Βαθμίδα**	**Υπογραφή**
1.	.....	.....
2.	.....	.....
3.	.....	.....

ABSTRACT

Simulations of distributed systems are powerful, since they allow us to prove the relations among different systems, compare their capabilities and performance and transfer positive and impossibility results from one to another. This thesis aims in analysing the equivalence of the message passing model and the mobile agent model via a simulation relation. Furthermore, it will examine the simulations that can be obtained between the message passing model and the mobile agent model, when some components of the systems may fail or be under adversarial attack.

We prove the following a) simulation relations of a mobile agent system with black holes by a message passing system with always dead processes, b) a simulation of a message passing algorithm that sends at most $k-1$ messages to always dead processes by a mobile agent system with black holes and at least k mobile agents, c) a simulation of mobile agent black$^+$ hole system by a message passing system with crash failures and d) a simulation of mobile agent gray hole system by a message passing system with omission failures. Furthermore we present some positive and impossibility results as a consequence of the simulations presented.

ΣΥΝΟΨΗ

Οι προσομοιώσεις στα κατανεμημένα συστήματα είναι πολύ ισχυρά εργαλεία, καθώς μας επιτρέπουν να αποδεικνύουμε τις σχέσεις διαφορετικών συστημάτων, να συγκρίνουμε τις δυνατότητες και την απόδοση τους και να μεταφέρουμε θετικά και αρνητικά αποτελέσματα από ένα σύστημα στο άλλο. Σε αυτή την εργασία θα εξετάσουμε την ισοδυναμία των κατανεμημένων συστημάτων ανταλλαγής μηνυμάτων (MP) και κινητών πρακτόρων (MA) μέσω μίας σχέσης προσομοίωσης. Στη συνέχεια θα ερευνηθεί ποιες προσομοιώσεις μπορούν να αποδειχθούν ανάμεσα στο μοντέλο ανταλλαγής μηνυμάτων και στο μοντέλο κινητών πρακτόρων, όταν κάποιες συνιστώσες των συστημάτων υποστούν βλάβη ή λειτουργούν υπό τις εντολές κάποιου αντιπάλου.

Θα αποδείξουμε τις ακόλουθες σχέσεις προσομοίωσης: α) συστημάτων MA με μαύρες οπές από συστήματα MP με "διαρκώς αδρανείς επεξεργαστές", β) αλγορίθμων MP που στέλνονται το πολύ $k-1$ μηνύματα σε "διαρκώς αδρανείς επεξεργαστές" από συστήματα MA με μαύρες οπές και τουλάχιστον k πράκτορες, γ) συστημάτων MA με μαύρες$^+$ οπές από συστήματα MP με χαλασμένους επεξεργαστές και δ) προσομοίωση συστημάτων MA με γκρίζες οπές από συστήματα MP με επεξεργαστές που παραλείπουν βήματα του αλγορίθμου. Επιπλέον, θα παρουσιάσουμε αποτελέσματα και αποδείξεις αδυναμίας επίτευξης στόχου των συστημάτων ανταλλαγής μηνυμάτων και κινητών πρακτόρων που προκύπτουν ως απόρροια των προσομοιώσεων που αποδείχθηκαν.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Αριστείδη Παγουρτζή, για την σωστή καθοδήγηση του, τις συμβουλές, ιδέες και γνώσεις του και την υπομονή του κατά την εκπόνηση της διπλωματικής μου εργασίας. Επιπλέον θα ήθελα να ευχαριστήσω τα μέλη της τριμελούς επιτροπής, κ. Δημήτριο Φωτάκη και κ. Ευστάθιο Ζάχο για το χρόνο τους και για τις γνώσεις που μου μεταλαμπάδευσαν με τη διδασκαλία τους.

Επίσης θα ήθελα να ευχαριστήσω τους συμφοιτητές και φίλους μου για τη βοήθεια και τη στήριξη που μου παρείχαν μέσα από τις συζητήσεις μας.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, για την αγάπη τους, τη στήριξη τους, την εμπιστοσύνη τους και την αμέριστη υπομονή τους καθ' όλα τα χρόνια των σπουδών μου.

CONTENTS

CHAPTER 1

DISTRIBUTED COMPUTING MODELS

1.1 Introduction

Distributed Computing Models and distributed algorithms are designed to solve problems when a distributed or dislocated set of processes want to achieve a certain common goal, under various assumptions about the communication of the processes, the mobility of the processes as well as the possible failures of the communication network and of the processes. Distributed algorithms have many applications in real-life scenarios and the research goal is to design realistic distributed computing models that capture various attacks.

In distributed computing, two main models are the message passing model and the mobile agent model. In the message passing model, there are n stationary processes that communicate to each other through some communication channels, by exchanging messages. The message passing model can be represented by a directed or undirected graph $G = (V, E)$, where the vertices correspond to the processes and the edges to the communication channels. In the mobile agent model, there are k non stationary processes, called agents, that move to different execution places, according to the provided navigation subsystem.

Chalopin et al proved in [9] that every message passing algorithm can be simulated in the mobile agent system, and vice versa, given that the navigation system of the mobile agent system and is a graph $G = (V, E)$ where the vertices represent the execution places and the edges the navigation links is equal with the communication system of the message passing model. This paper established the equivalence between the message passing model and the mobile agent model.

Simulations are useful for proving impossibility results and solving problems between different models of computation. Many major results in distributed computing were established using simulation relations. For example it was proved in [2] that any task computable with less than $|V|/2$ faulty processes in the message passing model is computable in the shared memory model and vice versa, using emulators, that translate algorithms from the shared memory model to the message passing model. Furthermore, the impossibility of the k-set consensus problem, which is a generalization of the impossibility of the consensus problem (FLP [17]) was proved by simulation, by the "BG-distributed simulation algorithm", which was introduced in [7] and proved in [8].

The "BG simulation algorithm" transforms any k-fault tolerant solution for the k-set consensus into a $k+1$ wait free solution in the shared memory model. Since the latter is impossible, the simulation proves the impossibility of the k-set consensus.

The problem of Byzantine Synchronous consensus can be solved by combining a series of simulations from Byzantine to "identical" Byzantine, from "identical" Byzantine to omission failure and from omission failure to crash failure model in $4(f+1)$ rounds when $n > 4f$, where f is the number of Byzantine failures, as presented in [4, pp. 251–275], [26]. Although the Byzantine Agreement problem can be solved optimally in $t+1$ rounds, when $n > 3f$ [18], the solution using the simulations despite not being optimal, it is much simpler.

In this thesis we will focus on simulation relations between Mobile Agent and Message Passing Algorithms. In chapter 1 we will define the Message Passing (MP) and the Mobile Agent (MA) model and generalize the definitions of simulations of [23] and [4]. In chapter 2 we will present and analyse the simulations of [6, 9] both in the asynchronous and the synchronous model. In chapter 3 we will present the simulation of [13] of a Message Passing algorithm by Mobile Agent System where agents may crash and examine if there are simulation relations between message passing and mobile agent systems, when some components of the two systems, real and simulated, may fail or behave maliciously. More specifically, we prove the following: a) a simulation of mobile agent system with black holes by a message passing system with always dead processes, b) a simulation of message passing algorithm that sends at most $k-1$ messages to always dead processes by a mobile agent system with black holes and at least k mobile agents, c) a simulation of mobile agent black$^+$ hole system by a message passing system with crash failures and d) a simulation of mobile agent gray hole system by a message passing system with omission failures. Lastly, in chapter 4 we will present some use cases of the simulations presented in chapter 3, where positive results and impossibility results of one system will be transformed into positive and impossibility results for the other system.

1.2 Message Passing Model (MP)

In order to define a message passing system we will use the definition and notation of message passing systems of [32, pp. 43–47], [9], [22, pp. 39–49], [4, pp. 9–15]. In distributed computing, a message passing system $(P,\ C, \lambda)$ is formed by a collection P of processes that communicate to each other by exchanging messages through the communication subsystem C and each process $p \in P$ has an initial state $\lambda(p)$ that encodes the initial input of the process i.e. the ID of the process (an integer in case of non anonymous network and $\perp$ otherwise), topology information, sense of direction e.t.c. The communication subsystem is represented as a simple connected undirected graph $G = (V,\ E,\ \delta)$ where:

- Each process $p \in P$ is located on a vertex of V. ($|\mathbb{P}| = |V| = n$)
- Edges E represent the communication channels between the processes.
- $\delta_u : N_{_G}(u) \to [1,\ deg_{_G}(u)]$ is the port labelling function, where $N_{_G}(u) = \{v \in V \mid \{u,\ v\} \in E\}$ and $deg_{_G}(u) = |N_{_G}(u)|$, and labels the edges incident to every vertex $u \in V$.

Processes communicate to each other by sending and receiving messages through the communication channels. Processes know through which port a message was re-

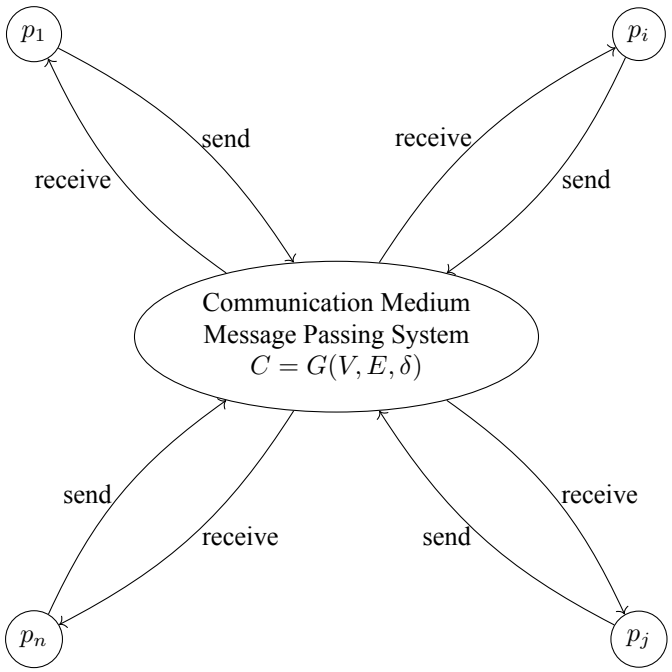

Figure 1.1: Communication of processes $p_1, ..., p_n$ through a message passing communication system $G(V, E, \delta)$

ceived and can choose through which port a message will be sent. The communication of processes through a message passing communication system is illustrated in figure 1.1, where the communication system is responsible for delivering the send events of processes to the corresponding port/recipient of the receive event [29, pp. 3–7].

The main models of communication of processes are the *LOCAL*, the *CONGEST* and the *ASYNC* [27, pp. 15–29].In the *LOCAL* model, communication of processes takes place within **synchronous** rounds and processes wake up simultaneously and start the computation at the same round. It aims in capturing and examining the locality of distributed computing. For this reason it allows exchanging messages of unlimited size and unlimited local computations, avoiding the obstacles imposed by congestion and asynchrony.

The *CONGEST* model, focuses mostly in the volumn of communication and in the time and message complexity of the protocol. In this model the maximum message size is $O(\log n)$ bits and the communication can be either synchronous or asynchronous.

In the *ASYNC* (asynchronous) model, the communication of processes is **asynchronous** and receiving a message from a neighboring process takes finite but unpredictable time. Therefore this model aims in capturing and examining the effects of asynchrony.

In this thesis we will consider mostly the *ASYNC* and the *LOCAL* model.

Message Passing algorithm

Processes in the message passing systems can be modelled as I/O automata [24, 23], state machines [27, 4] or transition systems [32]. Following the approach of [32], each

process in the message passing system is a transition system, that can interact with the communication subsystem. Let **M** denote the set of all possible messages. For each process $p \in P$ the local algorithm $\mathcal{D}_p$ is defined by:

- the set of possible states Q (not necessarily finite)
- the set of initial states $I \subseteq Q$
- the initial state $\lambda(p) \in I$ of $p \in P$
- a relation $\vdash_p$ of events (internal events, send events or receive events)

Let $state_i(p),\ i \geq 0$ be the state of process $p \in P$. Let M be the multiset of messages in transit. M is initially empty. A message in transit is a triple (p, m, p') where $p \in P$ is the sender, $m \in \mathbf{M}$ is the message and $p' \in P$ is the receiver. [1] The set of possible events of message passing algorithm $\mathcal{D}$ is denoted as $events(\mathcal{D})$. The possible events associated to process $p \in P$ are either:

- **internal events**
$$(c,\ 0,\ \perp) \vdash_p (d,\ 0,\ \perp)$$
where c is the old state of p and d is the new state of p.
$\perp$ is the null message and 0 indicates that no message is sent nor received.
- **send events**
$$(c,\ 0,\ \perp) \vdash_p (d,\ out,\ m)$$
where c is the old state of p and d is the new state of p after sending message m through port out.
$M \leftarrow M \cup \{(p, m, p')\}$, where $p' \in P$ is such that $\delta_p(p') = out$
- **receive events**
$$(c,\ in,\ m) \vdash_p (d,\ 0,\ \perp)$$
where c is the old state of p and d is the new state of p after receiving message m through port in.
$M \leftarrow M \setminus \{(p', m, p)\}$, where $p' \in P$ is such that $\delta_p(p') = in$

A distributed Message Passing algorithm $\mathcal{D}$ for a collection of processes $P = \{p_1, p_2, ..., p_n\}$ is the collection of local algorithms $\mathcal{D}_p,\ p \in P$. A transition in a message passing algorithm is defined by an event on a process. A configuration of $\mathcal{D}$ is defined as $config_i(\mathcal{D}) = (state_i, M_i)$, where $state_i(p), p \in P$ is the state of process p and M_i is the multiset of messages in transit. Initially $state_0(p) = \lambda(p) \in I\ , p \in P$ and $M_0 = \emptyset$. Therefore, the execution $\mathcal{E}$ of the message passing algorithm $\mathcal{D}$ is defined as a sequence of configurations and events:

$$\mathcal{E} = (state_i,\ M_i, \phi_i)_{i \geq 0}$$

where for each i there exist a unique process $p \in P$ such that:

- $state_{i+1}(p') = state_i(p'),\ \forall p' \in P : p' \neq p$
- $state_{i+1}(p)$ and M_{i+1} are obtained from $state_i$ and M_i by the applicable event ϕ_i on p.

A configuration is **terminal** when there are no applicable events.

[1] Processes do not necessarily have unique identifiers

Termination

A process is active if an internal or send event is applicable in his state and passive otherwise. [32] A message passing algorithm $\mathcal{D}$ terminates:

- **Implicitly:** when all processes are passive and there are no messages in transit.
- **Explicitly:** if it terminates implicitly and at least one process detects the termination of the algorithm, in the sense that all processes have computed their final values.

Complexity Measures

In a message passing algorithm the complexity measures we are interested in analysing are the number of messages exchanged, the amount of time for the termination (implicit or explicit) of the algorithm and the space complexity in terms of total or local memory requirements.

Definition 1.1. The **message complexity** [4] or **communication complexity** [23] of an algorithm $\mathcal{D}$ on a communication system C, denoted as $MSG(\mathcal{D}, C)$, is the maximum number of non-null messages exchanged in the message passing algorithm over all executions.

Definition 1.2. The **bit complexity** [23]an algorithm $\mathcal{D}$ on a communication system C, denoted as $Bit(\mathcal{D}, C)$, is defined as the total number of bits in the messages exchanged in the message passing algorithm over all executions.

Definition 1.3. The **time complexity** [4] or **round complexity** on a **synchronous** system of an algorithm $\mathcal{D}$ on a communication system C, denoted as $Time(\mathcal{D}, C)$, is the maximum number of rounds until termination over all executions of the message passing algorithm.

In asynchronous systems, defining the time complexity requires more attention, since several delays may occur at the processes. The delay of a message is defined as the time that elapses between a send event and the corresponding receive event.

Definition 1.4. The **time complexity** of an **asynchronous** system [4, 27], of an algorithm $\mathcal{D}$ on a communication system C, denoted as $Time(\mathcal{D}, C)$, is the maximum number of time units from the start of the execution of algorithm $\mathcal{D}$ until the termination, over all executions of the message passing algorithm $\mathcal{D}$.

Another time complexity measure, namely the **round complexity** of **asynchronous** systems, was defined in [11] in order to capture the notion of round complexity of synchronous systems but in asynchronous systems. In this work, the delay value T is assumed to be chosen by an adversary and the definition of an asynchronous round for process p_i is the number of times that p_i alternated between receive and send events.

Definition 1.5. The **asynchronous round complexity** [11] of an algorithm $\mathcal{D}$ on a communication system C, denoted as $RC(\mathcal{D}, C)$, is the maximum number of rounds of all processes over all executions of $\mathcal{D}$.

Definition 1.6. The **total space complexity** of an algorithm $\mathcal{D}$ on a communication system C, denoted as $Mem(\mathcal{D}, C)$, is the total number of memory bits used by the algorithm in the network in the worst case. The **local space complexity** or maximum space complexity of an algorithm $\mathcal{D}$ on a communication system C, denoted as $LocMem(\mathcal{D}, C)$, is the maximum number of memory bits use by the algorithm at any processor in the network in the worst case. [27].

1.3 Mobile Agent Model (MA)

In order to define a mobile agent system we will use the definition and notation of message passing systems of [9]. In the mobile agent model there is a set $\mathbb{A}$ of k identical mobile agents, which are non stationary computational entities with internal memory, called notebook. Mobile agents are located on execution places $\mathbb{P}$ and can move from place to place through the navigation subsystem $\mathbb{S}$ and execute the places that they visit and read and write data on the local memory of the execution places. Mobile agents can be:

- **synchronous**, where traversing one edge of the navigation subsystem and executing a place takes places within a round.
- **asynchronous**, where each of their actions is executed in finite but unpredictable time.

The navigation subsystem $\mathbb{S}$ is a simple undirected connected labelled graph $G = (V, E, \delta)$, where agents are located on the vertices V and can move through the links E.

- The vertices V represent the execution places $\mathbb{P}$, and are equipped with a whiteboard, which is local memory that agents can access.
- The edges incident to every vertex $u \in V$ are distinctly labelled according to the port labelling function $\delta_u : N_G(u) \rightarrow [1, deg_G(u)]$,
  where $N_G(u) = \{v \in V \mid \{u, v\} \in E\}$ and $deg_G(u) = |N_G(u)|$
  When an agent migrates from a place knows through which port it migrates.

The function λ describes the initial states. More precisely, $\lambda(\alpha)$, $\alpha \in \mathbb{A}$ is the initial state of mobile agent α and $\lambda(p)$, $p \in \mathbb{P}$ is the initial state of execution place p, which encodes the initial input of the process i.e. the ID of the process or the execution place (an integer in case of non-anonymous network and $\perp$ otherwise), topology information, sense of direction e.t.c. Agents are initially located on the vertices of G according to the function $\pi_0 : \mathbb{A} \rightarrow V$, and for each agent $\alpha \in \mathbb{A}$ the initial location $\pi_0(\alpha)$ is called *homebase* of agent α. An agent $\alpha \in \mathbb{A}$ located at an execution place $p_i \in \mathbb{P}$ can copy $whiteboard_{p_i}$ to $notebook_\alpha$, perform (local) computations, write on $whiteboard_{p_i}$, depart from p_i to $p_k \in N_G(p_i)$ and arrive to p_k. The navigation of mobile agents through the navigation system is illustrated in figure 1.2, where agents can depart from an execution place p_i to migrate to execution place p_k and arrive to execution place p_i from execution place p_k using the navigation system $\mathbb{S} = G(V, E, \delta)$.

Therefore the mobile agent system is described as

$$(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$$

.

Mobile Agent Algorithm

Following the definition of mobile algorithm of [9] each agent in the mobile agent system is a transition system, that can interact with the execution places and the navigation subsystem. For each mobile agent $\alpha \in \mathbb{A}$ the local algorithm $\mathcal{A}_\alpha$ is defined by:

- the set of possible states $Q_\mathbb{P}$ of the execution places(not necessarily finite).

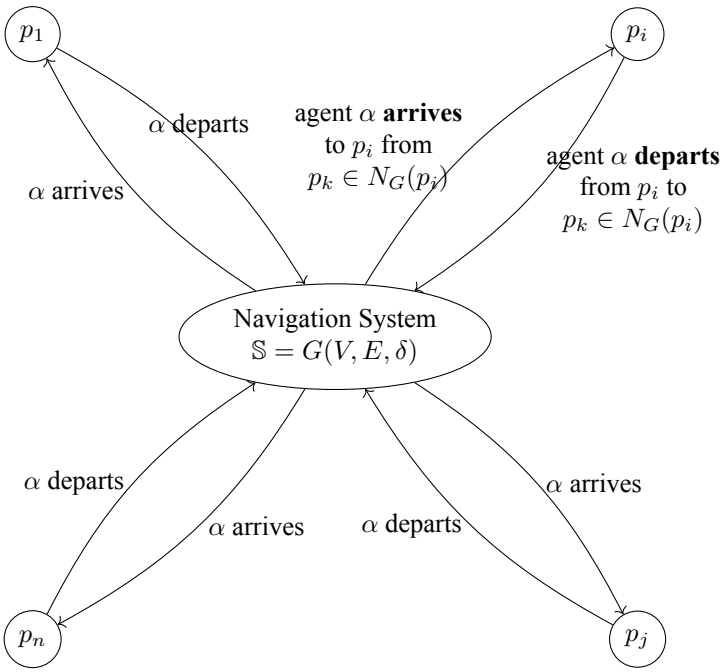

Figure 1.2: Navigation of mobile agent $\alpha \in \mathbb{A}$ over the execution places $p_1, ..., p_n$ through a mobile agent navigation system $G(V, E, \delta)$

- the set of possible states $Q_{\mathbb{A}}$ of the agents (not necessarily finite).
- the set $I_{\mathbb{A}} \subseteq Q_{\mathbb{A}}$ of initial states $\lambda(\alpha) \in I_{\mathbb{A}}$, of agents $\alpha \in \mathbb{A}$
- the set $I_{\mathbb{P}} \subseteq Q_{\mathbb{P}}$ initial states $\lambda(p) \in I_{\mathbb{P}}$ of execution places $p \in \mathbb{P}$
- a relation $\vdash_p^\alpha$ of events.

Let $state^{\mathcal{A}}(\alpha)$ be the state of agent $\alpha \in \mathbb{A}$ and $state^{\mathcal{A}}(p)$ be the state of execution place $p \in \mathbb{P}$. Let $\mathbb{M}$ be the set of mobiles agents in transit. $\mathbb{M}$ is initially empty. An agent in transit is a triple (p, α, p') where $p \in \mathbb{P}$ is the place of departure and $p' \in \mathbb{P}$ is the place of arrival. [2] Let $\pi : \mathbb{A} \to \mathbb{P}$ be a mapping of agents that are not in transit. The set of possible events is denoted as $events(\mathcal{A})$. The possible events associated to agent $\alpha \in \mathbb{A}$ are:

- α **departs from** p

  $$(s,\ q,\ 0) \vdash_p^\alpha (s',\ q',\ out)$$

  where s is the old state of α, s' is the new state of α, q is the old state of p and q' is the new state of p, after the departure of α through port out.
  $\mathbb{M} \leftarrow \mathbb{M} \cup \{(p, \alpha, p')\}$, where $p' \in \mathbb{P}$ is such that $\delta_p(p') = out$
  $\pi(\alpha) \leftarrow \perp$

- α **arrives at** p

  $$(s,\ q,\ in) \vdash_p^\alpha (s',\ q',\ 0)$$

[2] Execution places do not necessarily have unique identifiers

where s is the old state of α, s' is the new state of α, q is the old state of p and q' is the new state of p, after the arrival of α through port in.
$\mathbb{M} \leftarrow \mathbb{M} \setminus \{(p', \alpha, p)\}$, where $p' \in \mathbb{P}$ is such that $\delta_p(p') = in$
$\pi(\alpha) \leftarrow p$

- α **remains at** p

$$(s,\, q,\, 0) \vdash_p^{\alpha} (s',\, q',\, 0)$$

where s is the old state of α, s' is the new state of α, q is the old state of p and q' is the new state of p.

A Mobile agent algorithm $\mathcal{A}$ for a collection of mobile agents $\mathbb{A}$ is the collection of local algorithms $\mathcal{A}_\alpha,\ \alpha \in \mathbb{P}$. A configuration of $\mathcal{A}$ is defined as $config_i(\mathcal{A}) = (state_i^{\mathcal{A}}, \mathbb{M}_i, \pi_i)$, where $state_i^{\mathcal{A}}(\alpha), \alpha \in \mathbb{A}$ is the state of agent α, $state_i^{\mathcal{A}}(p),\ p \in \mathbb{P}$ is the state of execution place p, $\mathbb{M}_i$ is the multiset of agents in transit and π_i is the position of agents that are not in transit. Initially $state_0^{\mathcal{A}}(\alpha) = \lambda(\alpha), \alpha \in \mathbb{A}$, $state_0^{\mathcal{A}}(p) = \lambda(p), p \in \mathbb{P}$ and $\mathbb{M}_0 = \emptyset$. Therefore, the mobile agent algorithm is defined as

$$\mathcal{A} = (\vdash_p^{\alpha})_{\alpha \in \mathbb{A},\, p \in \mathbb{P}}$$

and the execution $\mathcal{E}$ of the mobile agent algorithm $\mathcal{A}$ is defined as the sequence of cnfigurations:

$$\mathcal{E} = (state_i^{\mathcal{A}},\, \mathbb{M}_i,\, \pi_i)_{i \geq 0}$$

where for each i there exist a unique agent $\alpha \in \mathbb{A}$ and a unique place $p \in \mathbb{P}$ such that:

- $state_{i+1}^{\mathcal{A}}(\alpha') = state_i^{\mathcal{A}}(\alpha'),\ \forall \alpha' \in \mathbb{A} : \alpha' \neq \alpha$
- $state_{i+1}^{\mathcal{A}}(p') = state_i^{\mathcal{A}}(p'),\ \forall p' \in \mathbb{P} : p' \neq p$
- $state_{i+1}^{\mathcal{A}}(p)$ and $\mathbb{M}_{i+1}$ are obtained from $state_i^{\mathcal{A}}$ and $\mathbb{M}_i$ by an event on p.

A configuration is terminal when there are no applicable events.

Termination: A mobile agent is passive if there are no applicable events to his state. [9]. A terminal configuration of mobile agent algorithm $\mathcal{A}$ is a configuration where all agents are passive. A mobile agent algorithm $\mathcal{A}$ terminates:

- **Implicitly:** if it reaches a terminal configuration, but no agent is aware of the termination of $\mathcal{A}$.
- **Explicitly:** if it reaches a terminal configuration and at least one agent detects the termination of $\mathcal{A}$.

Complexity Measures

The complexity of a mobile agent algorithm can be measured by the number of mobile agents used, by the time until the termination(implicit or explicit) of the algorithm, by the number of moves or steps taken by the mobile agents and the memory requirements of the notebook of the mobile agents and the whiteboard of the execution places.

Definition 1.7. The **agent complexity** of a mobile agent algorithm $\mathcal{A}$ on a navigation system $\mathbb{S}$, denoted as $Agents(\mathcal{A}, \mathbb{S})$, is the minimum number k of mobile agents of the mobile agent system such that algorithm $\mathcal{A}$ correctly computes the outputs of the problem on $\mathbb{S}$ with k mobile agents over all executions of $\mathcal{A}$. [25]

Definition 1.8. The **time complexity** of a mobile agent algorithm $\mathcal{A}$ on a navigation system $\mathbb{S}$, denoted as $Time(\mathcal{A}, \mathbb{S})$, is the maximum number of time units until the termination of the algorithm, over all executions of algorithm $\mathcal{A}$ [15, 12], assuming that each move(traversal of a link) of an agent takes one time unit. In the case of asynchronous mobile agent this is referred as ideal time [25].

Definition 1.9. The **move complexity** of an agent α of a mobile agent algorithm $\mathcal{A}$ on a navigation system $\mathbb{S}$, denoted as $Move(\alpha, \mathcal{A}, \mathbb{S})$, is the maximum number or moves (traversals of links) of agent $\alpha \in \mathbb{A}$, until the termination of the algorithm, over all executions of algorithm $\mathcal{A}$. The **total move complexity** of algorithm $\mathcal{A}$ on a navigation system $\mathbb{S}$ is $TotalMoves(\mathcal{A}, \mathbb{S}) = \sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S})$

We note that for the time complexity and the move complexity of a mobile agent algorithm $\mathcal{A}$ and a navigation system $\mathbb{S}$ of a mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$ the following relation holds: $Time(\mathcal{A}, \mathbb{S}) = \max_{\alpha \in \mathbb{A}} Move(\alpha, \mathbb{A}, \mathbb{S})$

The **space complexity** is measured in terms of the whiteboard memory and the notebook memory, and we are interesting in analysing the space complexity both locally, on each execution place or agent and globally for the entire system.

Definition 1.10. The **total space complexity of execution places** of a mobile agent algorithm $\mathcal{A}$ on a navigation system $\mathbb{S}$, denoted as $MemWB(\mathcal{A}, \mathbb{S})$, is the total number of memory bits used by execution places(whiteboards), during the execution of the algorithm $\mathcal{A}$ in the navigation system $\mathbb{S}$ in the worst case.

Definition 1.11. The **local space complexity of an execution place**, of a mobile agent algorithm $\mathcal{A}$ on a navigation system $\mathbb{S}$, denoted as $LocMemWB(\mathcal{A}, \mathbb{S})$, is the maximum number of memory bits used by algorithm $\mathcal{A}$ at any execution place(whiteboard) of the navigation system in the worst case.

Definition 1.12. The **total space complexity of a mobile agent** of a mobile agent algorithm $\mathcal{A}$ on a navigation system $\mathbb{S}$, denoted as $MemAg(\mathcal{A}, \mathbb{S})$, is the total number of memory buts used by mobile agents(notebooks) during the execution of algorithm $\mathcal{A}$ in the navigation system $\mathbb{S}$ in the worst case.

Definition 1.13. The **local space complexity of a mobile agent** of a mobile agent algorithm $\mathcal{A}$ on a navigation system $\mathbb{S}$, denoted as $LocMemAg(\mathcal{A}, \mathbb{S})$, is the maximum number of memory bits used by any mobile agent(notebook) on the execution of algorithm $\mathcal{A}$ on the navigation system in the worst case.

1.4 Simulations

A simulation between two distributed systems A_1 and A_2 is a relation between them, that indicates that they have the same input/output behavior, under the same conditions. An algorithm $\mathcal{A}_1$ of the communication system C_1 simulates algorithm $\mathcal{A}_2$ of the communication system C_2, if for every execution $\mathcal{E}_1$ of $\mathcal{A}_1$ there exist an execution $\mathcal{E}_2$ of $\mathcal{A}_2$ that corresponds to $\mathcal{E}_2$.

In [23, pp. 224–228] a simulation relation is defined between two I/O automata that have the same external interface. A similar definition of simulations is given in [4, pp. 157–166], where the simulation relation is defined between two communication systems. We will extend the definitions of simulation of [23] and [4] to define simulations between two distributed systems.

Definition 1.14. (extension of the definition of simulation of [23])
Let A_1, A_2 be two distributed systems, with corresponding communication systems C_1, C_2. Suppose f is a binary relation over $config(A_1)$ and $config(A_2)$:

$$f \subseteq config(A_1) \times config(A_2)$$

and h is an injective function $h : events(A_2) \rightarrow events(A_1)$. Then f is a simulation relation from A_1 to A_2 if the following hold:

1. If s is an initial configuration of A_1, then the intersection of $f(s)$ with the set of initial configurations of A_2 is non empty.

2. If s is a reachable configuration of A_1, $u \in f(s)$ is a reachable configuration of A_2 and π is an applicable event on s, that changes the configuration of A_1 into s', then there exists an execution segment $\mathcal{E}$ of A_2 starting with u and ending with $u' \in f(s')$ such that
$$h(trace(\mathcal{E})) = trace(\pi)$$
where $trace(\mathcal{E})$ is the subsequence of $\mathcal{E}$ consisting of all the external events.

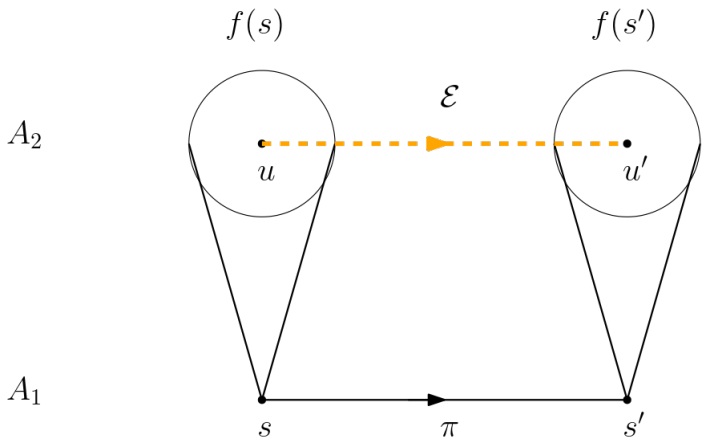

Figure 1.3: Simulation relation from system A_1 to A_2.
Any applicable event on a state of A_1 has a corresponding state and sequence of events in A_2, for which the resulting state of A_2 corresponds to the resulting state of A_1

The definition of simulation of [4] is based on the **Layered model**. The layering technique is often used to control the complexity of large scale systems. In the layered model we assume a stack of processes, one on top of the other, where each process in the stack interacts only with the layer above and below. The top process communicates with the external environment and the bottom with the communication system. A communication system C has a set of inputs, denoted as $in(C)$, a set of outputs, denoted as $out(P)$ and a set of allowable sequences $seq(C)$ of inputs and outputs. The sets of inputs and outputs form the **interface** of C. For an execution $\mathcal{E}$ we denote as $top(\mathcal{E})$ and $bot(\mathcal{E})$ the restriction of the sequence of events of execution $\mathcal{E}$ to the events of the top and respectively bottom interface.

Admissibility of an execution of a layered communication system (C_2, C_1) is a property that captures the proper behavior of the system. Informally, an execution of a layered system (C_2, C_1) is admissible if it does not halt while there is an applicable event, if the inputs of satisfy the input constraints of C_2 and the communication system is correct, according to the specifications of C_1. More formally, an execution α is $(C_2,\ C_1)$-admissible, if it is:

- *fair*, that is every event that is continuously enabled eventually occurs.
- *user compliant* for C_2, that is for every prefix $\alpha'\phi$ of execution α, where ϕ is an input event from the environment to C_2, if α' is a prefix of some element of $seq(C_2)$ then so is $\alpha'\phi$
- *correct* for C_1,that is if $bot(\alpha)$ is an element of $seq(C_1)$.

Example 1.15. The interface to an asynchronous message passing system has two types of events:

- **send event:** $(c, 0, \perp)\ \perp_p\ (d, out, m)$
  which is an input event of the message passing system on behalf of processor p, that sends message m through the communication port out.
- **receive event:** $(c, in, m)\ \perp_p\ (d, 0, \perp)$
  which is an output event of the message passing system on behalf of processor p, where message m is received through communication port in.

The interface of an asynchronous message passing system is depicted in figure 1.4

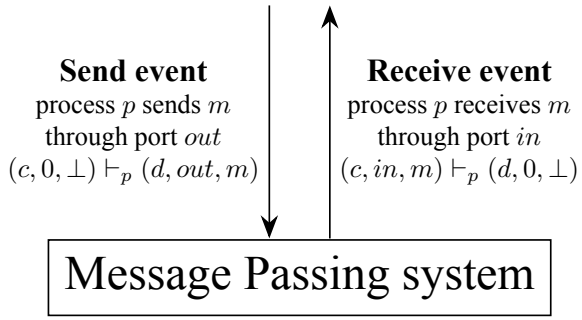

Figure 1.4: The interface of the asynchronous message passing system

Example 1.16. The interface to an asynchronous mobile agent system has two types of events:

- **depart event:** $(s, q, 0)\ \perp_p^{\alpha}\ (s', q', out)$
  which is an input event of the mobile agent system on behalf of agent $\alpha \in \mathbb{A}$ located at execution place p, that departs from p through the port out.
- **arrive event:** $(s, q, in)\ \perp_p\ (s', q', 0)$
  which is an output event of the mobile agent system on behalf of agent $\alpha \in \mathbb{A}$ located at execution place p, that arrives to p through the port in.

The interface of an asynchronous mobile agent system is depicted in figure 1.5

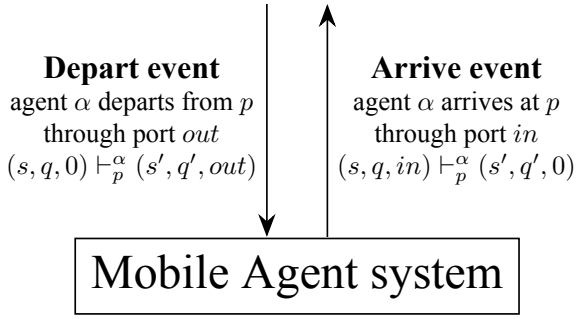

Figure 1.5: The interface of the asynchronous mobile agent system

Definition 1.17. (extension of the definition of simulation of [4].)
Communication system C_1 of distributed system A_1 simulates communication system C_2 of distributed system A_2 if there exists a collection of processes, called Sim that satisfies the following

1. The top interface of Sim is the interface of C_2.
2. The bottom interface of Sim is the interface of C_1.
3. For every (C_2, C_1)-admissible execution $\mathcal{E}$ of Sim, there exists a sequence σ of allowable inputs and outputs of C_2 such that $\sigma = top(\mathcal{E})$

Figure 1.6 illustrates the definition of simulation 1.17

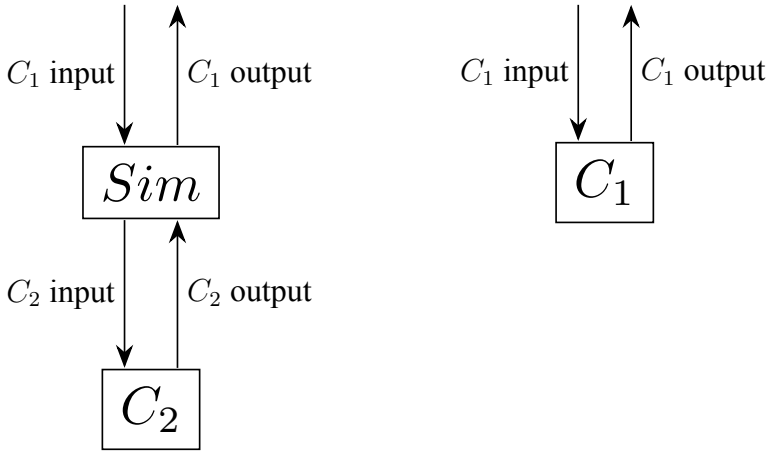

Figure 1.6: Simulation of system C_2 on top of C_1.
The inputs of the Simulation processes are transformed to inputs of system C_1 and the outputs of system C_1 are transformed to outputs of system C_2. Running the simulation on top of C_1 produces the same appearance as does in C_2

We note that definitions 1.14 and 1.17 are equivalent.

Proposition 1.18. The definition of simulation of 1.14 and 1.17 are equivalent

Proof. ($\Rightarrow$) Assume that f is a simulation relation, according to definition 1.14, from A_1 to A_2 and h is an injective function $h : events(A_2) \rightarrow events(A_1)$. Let Sim be a set of processors that has as top interface A_2, bottom interface A_1 and maps inputs, outputs and states of A_2 to A_1 according to f and h.

Let $\mathcal{E}$ be a (C_2, C_1)-admissible execution of Sim. Let $\mathcal{E}$ be of the form $(state_i, event_i)_{i\geq 0}$. Let $\mathcal{E}|A_1$ be the restriction of $\mathcal{E}$ to states and events of system A_1 and is of the form $\mathcal{E}|A_1 = (state'_i, event'_i)_{i\geq 0}$.

By definition 1.14 $\forall i\, u \in f(state'_i)$, $state'_i$ is reachable configuration of A_1, $event'_i$ is an applicable event on $state'_i$ that changes the configuration to $state'_{i+1}$. Hence by definition 1.14 there exists an execution segment α_i of A_2 starting with u and ending with $u' \in f(state'_{i+1})$ such that: $h(trace(\alpha_i)) = trace(\pi)$.

Let $\sigma = \alpha_0\, \alpha_1\, \alpha_2 \ldots$. We note that σ is an execution of A_2 for which $\sigma = top(\mathcal{E})$. This is illustrated in figure 1.4 Therefore C_1 simulates C_2 according to definition 1.17.

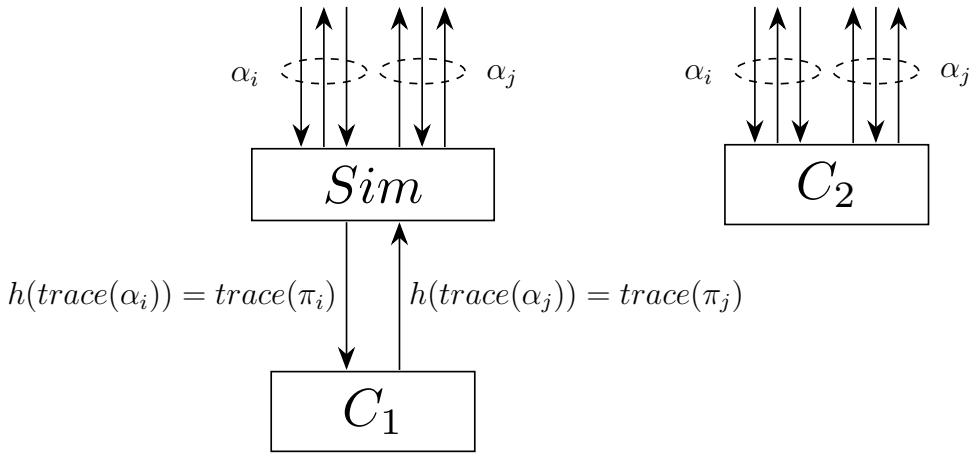

Figure 1.7: Sketch of the proof of the implication of definition 1.17 by 1.14

($\Leftarrow$) Assume that C_1 simulates C_2 according to definition 1.17 for a collection of processes Sim. Suppose that f is a binary relation that maps $f : config(A_1) \times config(A_2)$ and h is injective $h : events(A_2) \rightarrow events(A_1)$, and f and h are defined according to the transformations that are performed at the Sim processes and f is a simulation relation from A_1 to A_2 according to definition 1.14.

1. If $s \in I_{A_1} \Rightarrow f(s) \cup I_{A_2} \neq \emptyset$, where I_{A_1}, I_{A_2} are the initial states of A_1, A_2, since the bottom interface of Sim is C_2 and the top interface is C_1.

2. Let s be a reachable configuration of A_1, $u \in f(s)$ a reachable configuration of A_2 and π applicable event on s such that $(s, \pi) \vdash s'$.
   $\Rightarrow$ there exists a (C_2, C_1) admissible execution $\mathcal{E}$ of Sim where $s \in \mathcal{E}$ and π applicable event on s such that $(s, \pi) \vdash s'$.
   $\Rightarrow$ there exist sequence σ of allowable inputs and outputs of C_2 such that $\sigma = top(\mathcal{E})$. Hence there exists subsequence σ' of σ starting with $u \in f(s)$ ending with $u' \in f(s')$ such that $h(trace(\sigma')) = trace(\pi)$

□

CHAPTER 2

SIMULATIONS AMONG HONEST MA AND MP MODELS

2.1 Simulation of MA Algorithm by a MP System

It has been proved in [9] and [6] that every mobile agent algorithm can be simulated in the message passing model, by creating for each execution of a mobile agent algorithm $\mathcal{A}$ an equivalent execution in a message passing system.

Let $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$ a mobile agent system, where $\mathbb{S} = (V, E, \delta)$ is the navigation subsystem, and $\mathcal{A} = (\vdash_p^{\alpha})_{\alpha \in \mathbb{A},\, p \in \mathbb{P}}$.

The mobile agent algorithm $\mathcal{A}$ can be simulated in the message passing system $(P, C, \lambda') = (V, E, \delta, \lambda')$ where:

- for each execution place $\mathbb{P}$ of the mobile agent system there is a process $p \in P$ in the message passing system. Therefore there is a 1-1 correspondence between execution places and processes and $|P| = |\mathbb{P}|$. We will use the letter p to refer to both the process $p \in P$ in the message passing setting and for its corresponding execution place $p \in \mathbb{P}$ in the mobile agent setting.
- the communication subsystem C is equal to the navigation subsystem $\mathbb{S}$. $C = \mathbb{S} = (V, E, \delta)$
- the labelling of the processes, which encodes the initial state is:

$$\lambda'(v) = \begin{cases} (\lambda(v),\, 1, \lambda(\alpha)), & \text{if } v \text{ is the homebase of agent } \alpha \\ (\lambda(v), 0, \sharp), & \text{otherwise} \end{cases} \quad , v \in V$$

  where $\sharp$ is a null state.

Let $\mathcal{D}$ be the message passing algorithm that simulates the mobile agent algorithm $\mathcal{A}$. For each process $p \in P$ in the local simulation algorithm $\mathcal{D}_p$ we have:

- The set of possible states Q of a process is defined by the set of possible states $Q_{\mathbb{P}}$ of the execution places and by the state of a mobile agent if the mobile agent is present at the execution place.

$$Q \;=\; Q_{\mathbb{P}} \;\times\; \{0,\, 1\} \;\times\; Q_{\mathbb{A}} \cup \{\sharp\}$$

where $\{0,\ 1\}$ represents the presence of a mobile agent at an execution place and $\sharp$ is a null state and is used when there is no agent present at an execution place.

- The set of initial states $I \subseteq Q$ of a process is similarly defined as:

$$I\ =\ I_{\mathbb{P}}\ \times\ \{0,\ 1\}\ \times\ I_{\mathbb{A}} \cup \{\sharp\}$$

- The initial state $\lambda'(p) \in I$ of $p \in P$ is defined as:

$$\lambda'(p) = \begin{cases} (\lambda(p),\ 1,\ \lambda(\alpha)), \text{ if } p \text{ is the homebase of agent } \alpha \\ (\lambda(p),\ 0,\ \sharp), \text{ otherwise} \end{cases} ,p \in P$$

where $\lambda(p)$ is the initial state of the corresponding execution place $p\ \in\ \mathbb{P}$, 1 indicates that p is the homebase of some agent $\alpha\ \in\ \mathbb{A}$ and $\lambda(\alpha)$ is the is the initial state of agent $\alpha \in \mathbb{A}$ and 0 indicates that p is not homebase of any agent and $\sharp$ is the null state.

- $state(p) \in Q$, $p \in P$ is defined by the state of the corresponding execution place and the state of mobile agent $\alpha \in \mathbb{A}$ if α is located at the execution place $p \in \mathbb{P}$

$$state(p) = \begin{cases} (state^{\mathcal{A}}(p),\ 1,\ state^{\mathcal{A}}(\alpha)), \text{ if } \alpha \text{ is located at } p \\ (state^{\mathcal{A}}(p),\ 0,\ \sharp), \text{ otherwise} \end{cases} ,p \in P$$

- The presence of a mobile agent α at an execution place is encoded by a token $t(\alpha)$ and the state of the corresponding mobile agent is encoded by the value of the token, $t(\alpha) \in Q_{\mathbb{A}}$
- The set of possible messages $\mathcal{M} = Q_{\mathbb{A}}$ equals the set possible states of the agents.
- The mapping $h : events(\mathcal{A}) \rightarrow events(\mathcal{D})$ of possible events of the relation $\vdash_p^{\alpha}$ in the relation $\vdash_p$ of the message passing algorithm $\mathcal{D}$ is:

  A **departure event** $(s,\ q,\ 0)\ \vdash_p^{\alpha}\ (s',\ q',\ out)$, $p\ \in\ \mathbb{P}$, $\alpha\ \in\ \mathbb{A}$ is simulated in a **send event** $(c,\ 0,\ \perp)\ \vdash_p\ (d,\ out,\ m)$ for the corresponding process $p\ \in\ P$ where:

  - the old state c of p is: $c = (q,\ 1,\ s)$
  - the new state d of p is: $d = (q',\ 0,\ \sharp)$
  - the message $m = t(\alpha) = s'$ is sent through the port out and indicates that token $t(\alpha)$ is sent via port out.
    $M \leftarrow M \cup \{(p, m, p')\}$, where $p' \in P$ is such that $\delta_p(p') = out$

  An **arrival event** $(s,\ q,\ in)\ \vdash_p^{\alpha}\ (s',\ q',\ 0)$, $p\ \in\ \mathbb{P}$, $\alpha\ \in\ \mathbb{A}$ is simulated in a **receive event** $(c,\ in,\ m)\ \vdash_p\ (d,\ 0,\ \perp)$ for the corresponding process $p\ \in\ P$ where:

  - the old state c of p is: $c = (q,\ 0,\ \sharp)$
  - the new state d of p is: $d = (q',\ 1,\ s')$
  - the message $m\ =\ t(\alpha)\ =\ s$ is received through the port in and indicates that token $t(\alpha)$ is received via port in.
    port in.
    $M \leftarrow M \setminus \{(p', m, p)\}$, where $p' \in P$ is such that $\delta_p(p') = in$

An event where agent $\alpha \in \mathbb{A}$ **remains** at $p \in \mathbb{P}$ $(s,\, q,\, in) \vdash_p^{\alpha} (s',\, q',\, 0)$, is simulated in an **internal event** $(c,\, 0,\, \perp) \vdash_p (d,\, 0,\, \perp)$ for the corresponding process $p \in P$ where:

- the old state c of p is: $c = (q,\, 1,\, s)$
- the new state d of p is: $d = (q',\, 1,\, s')$

Let $\mathcal{D}_p$ be the algorithm induced by this construction on the process $p \in P$ and $\mathcal{D} = (\mathcal{D}_p)_{p \in P}$. The execution $\mathcal{E}'$ of the simulation algorithm $\mathcal{D}$ is the sequence:

$$\mathcal{E}' = (state_i,\, M_i)_{i \geq 0}$$

where for each i exists a unique process p such that:

- $state_{i+1}(p') = state_i(p'),\ \forall p' \in P\ :\ p' \neq p$
- $state_{i+1}(p)$ and M_{i+1} are obtained from $state_i$ and M_i by and event on p.

Proposition 2.1. [9] Let $\mathcal{A} = (\vdash_p^{\alpha})_{a \in \mathbb{A},\, p \in \mathbb{P}}$ be a mobile agent algorithm implemented on the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{S},\, \pi_0,\, \lambda)$. Let $\mathcal{D} = (\mathcal{D}_p)_{p \in P}$ be the message passing algorithm defined above on the message passing system $(P,\, C,\, \lambda')$.
Then algorithm $\mathcal{D}$ simulates algorithm $\mathcal{A}$ in the message passing system (P, C, λ')

Proof. Let $\mathcal{E}_{\mathcal{D}}$ be an execution of algorithm $\mathcal{D}_p$, $p \in P$ in the message passing system $(P,\, C,\, \lambda')$, f the simulation relation and h the events' mapping described above. By the construction of algorithm $\mathcal{D}_p$ from algorithm $\mathcal{A}$ of the system $(\mathbb{A},\, \mathbb{P},\, \mathbb{S},\, \pi_0,\, \lambda)$ we note that:

1. $f(\lambda') \cap I_{\mathbb{A}} \neq \emptyset$
2. If $s = (state_i, M_i)$, for some $i \in \mathbb{N}$ is a reachable configuration of $\mathcal{D}$, $u \in f(s)$, $u = (state_j^{\mathcal{A}}, \mathbb{M}_j, \pi_j), \in \mathbb{N}$ is a reachable configuration of $\mathcal{A}$ and π is an applicable event on s that changes the state from s to $s' = (state_{i+1}, M_{i+1})$ then by the mapping of events described above we note that there exists an event π_i' of the mobile agent system such that $h(\pi') = \pi$.

   Therefore, for any applicable event π we have that there exists an execution segment $\mathcal{E}_{\mathcal{A}}$ of $\mathcal{A}$ such that: $h(trace(\mathcal{E}_{\mathcal{A}})) = \pi$

Hence, by definition 1.14 the message passing system $(P,\, C,\, \lambda')$ simulates the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{S},\, \pi_0,\, \lambda)$ □

Termination

Lemma 2.2. Let $\mathcal{A} = (\vdash_p^{\alpha})_{a \in \mathbb{A},\, p \in \mathbb{P}}$ be a mobile agent algorithm implemented on the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{S},\, \pi_0,\, \lambda)$. Let $\mathcal{D} = (\mathcal{D}_p)_{p \in P}$ be the message passing algorithm defined above on the message passing system $(P,\, C,\, \lambda')$. If algorithm $\mathcal{A}$ has the termination property then algorithm $\mathcal{D}$ has the termination property.

Proof. Algorithm $\mathcal{D}$ simulates algorithm $\mathcal{A}$, therefore for every execution $\mathcal{E}_{\mathcal{D}}$ of $\mathcal{D}$ there exists an equivalent execution $\mathcal{E}_{\mathcal{A}}$ of $\mathcal{A}$.

If algorithm $\mathcal{A}$ terminates implicitly then all agents eventually become passive in $\mathcal{E}_{\mathcal{A}}$ and $\forall \alpha \in \mathbb{A}$ located at $p \in \mathbb{P}$ there is no applicable event of the relation $\vdash_p^{\alpha}$. By

the mapping of events in the simulation message passing algorithm $\mathcal{D}$ it is implied that $\forall p \in P$ there is no applicable event of the relation $\vdash_p$ on the execution $\mathcal{E}_{\mathcal{D}}$. Hence $\mathcal{D}$ terminate implicitly.

If algorithm $\mathcal{A}$ terminates explicitly then at least one agent $\alpha \in \mathbb{A}$ at $p \in \mathbb{P}$ detects termination in $\mathcal{E}$; i.e. that all execution places have their final values. By the construction of $\mathcal{D}$, process $p \in P$ with token $t(\alpha)$ detects termination ; i.e. that all processes have their final values. Hence $\mathcal{D}$ terminates explicitly. □

Remarks: In the simulation algorithm $\mathcal{D}$ described above a process $p \in P$ has an internal or send or receive event if and only if there is an agent on the corresponding execution place $p \in \mathbb{P}$. Hence during the execution of $\mathcal{D}$ there are at most k processes that execute events simultaneously, where $k = |\mathbb{A}|$.

By the construction of the simulation algorithm $\mathcal{D}$ there is a correspondence between the $whiteboard_p$, $p \in \mathbb{P}$ and the local memory of the corresponding process $p \in P$ and between $notebook_\alpha$, of mobile agent $\alpha \in \mathbb{A}$ and the messages exchanged in the message passing algorithm $\mathcal{D}$.

Complexity

Message complexity

Proposition 2.3. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm implemented on the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{S},\, \pi_0,\, \lambda)$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system $(P,\, C,\, \lambda')$, where $C = \mathbb{S}$. Then the **total number of messages exchanged**(message complexity) during the execution of $\mathcal{D}$ is:

$$MSG(\mathcal{D}, C) = \sum_{\alpha\in\mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) = TotalMove(\mathcal{A}, \mathbb{S})$$

where $Move(\alpha, \mathcal{A}, \mathbb{S})$ is the move complexity of agent $\alpha \in \mathbb{A}$ in the execution of $\mathcal{A}$. The **total size of messages exchanged** is

$$Bit(\mathcal{D}, C) = \sum_{\alpha\in\mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$$

where $LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$ is the local space complexity of agent α i.e. the size of notebook of α.

Proof. If in the simulation algorithm $\mathcal{D}$ there is a send event for process $p \in P$ and token $t(\alpha)$, $\alpha \in \mathbb{A}$ then in algorithm $\mathcal{A}$ agent $\alpha \in \mathbb{A}$ departs from the execution place $p \in \mathbb{P}$. Therefore, the number of times the token $t(\alpha)$ will be sent through the communication system is $Move(\alpha, \mathcal{A}, \mathbb{S})$ and hence the message complexity of $\mathcal{S}$ is $MSG(\mathcal{D}, C) = \sum_{\alpha\in\mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S})$. The size of each message $t(\alpha)$ in $\mathcal{D}$ equals the size of the $notebook_\alpha$ of the corresponding agent α or $state(\alpha)$, whose space complexity is $LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$. Therefore the bit complexity of the algorithm is: $Bit(\mathcal{D}, C) = \sum_{\alpha\in\mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$. □

Time Complexity

Proposition 2.4. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm implemented on the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{S},\, \pi_0,\, \lambda)$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing

algorithm defined above on the message passing system (P, C, λ'), where $C = \mathbb{S}$. Then the **time complexity** of $\mathcal{D}$ is:

$$Time(\mathcal{D}, C) = Time(\mathcal{A}, \mathbb{S})$$

where $Time(\mathcal{A}, \mathbb{S})$ is the time complexity of mobile agent algorithm $\mathcal{A}$ on the navigation system $\mathbb{S}$.

Proof. For every event of an execution $\mathcal{E}$ of $\mathcal{D}$ that involves token $t(\alpha), \alpha \in \mathbb{A}$, in the corresponding equivalent execution $\mathcal{E}'$ of $\mathcal{A}$ there is exactly one event that involves agent α, since each path traversal in $\mathcal{A}$ and each message delivery in $\mathcal{D}$ takes one unit of time. Therefore, the time complexity of $\mathcal{A}$ equals the time complexity of $\mathcal{D}$. □

Space Complexity

Proposition 2.5. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm implemented on the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system (P, C, λ'), where $C = \mathbb{S}$. Then the **local space complexity** of $\mathcal{D}$ is:

$$LocMem(\mathcal{D}, C) = \max_{p\in\mathbb{P}} \{LocMemWB(p, \mathcal{A}, \mathbb{S})\} + \max_{\alpha\in\mathbb{A}} \{LocMemAg(\alpha, \mathcal{A}, \mathbb{S})\}$$

where $LocMemWB(p, \mathcal{A}, \mathbb{S})$ is the local space complexity of execution place $p \in \mathbb{P}$ and $LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$ is the local space complexity of agent $\alpha \in \mathbb{A}$ of algorithm $\mathcal{A}$ on the navigation system $\mathbb{S}$.
The **total space complexity** of algorithm $\mathcal{D}$ is:

$$Mem(\mathcal{D}, C) = \sum_{p\in\mathbb{P}} LocMemWB(p, \mathcal{A}, \mathbb{S}) + \sum_{\alpha\in\mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$$

Proof. In the simulation algorithm $\mathcal{D}$ each process $p \in P$ stores and processes the contents of $whiteboard_p$ and $notebook_\alpha$, $\forall\alpha \in \mathbb{A}$. Therefore the local memory requirements for each process $p \in P$ is: $LocMem(\mathcal{D}, C) = \big(\max_{p\in\mathbb{P}}\{LocMemWB(p, \mathcal{A}, \mathbb{S})\} + \max_{\alpha\in\mathbb{A}}\{LocMemAg(\alpha, \mathcal{A}, \mathbb{S})\}\big)$ where $LocMemWB(p, \mathcal{A}, \mathbb{S})$ is the space complexity of the $whiteboard_p$ execution place p and $LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$ is the space complexity of $notebook_\alpha$ of agent $\alpha \in \mathbb{A}$ of algorithm $\mathcal{A}$. □

Table 2.1 summarises the simulation of a mobile agent algorithm $\mathcal{A}$ by message passing algorithm $\mathcal{D}$, table 2.2 summarises the complexity of the simulation and figure 2.1 illustrates the simulation relation.

Table 2.1: Simulation of Mobile Agent Algorithm $\mathcal{A}$ by Message Passing Algorithm $\mathcal{D}$

Mobile Agent Model	**Message Passing Model**
System: $(\mathbb{A},\ \mathbb{P},\ \mathbb{S},\ \pi_0,\ \lambda)$	**System:** $(P,\ C,\ \lambda')$
$\mathbb{S} = (V,\ E,\ \delta)$ navigation subsystem $\mathbb{P}$ or V : execution places, equipped with a whiteboard E : migration ports $\delta_u : N_G(u) \to [1,\ deg_G(u)] :\ u \in V$ port labelling function λ : initial states of places and agents	$C = (V,\ E,\ \delta)$ communication subsystem P or$V$: asynchronous processes E : communication channels $\delta_u : N_G(u) \to [1,\ deg(u)] :\ u \in V$ port labelling function λ' : initial states of processes $\lambda'(p) = \begin{cases} (\lambda(p),\ 1,\ \lambda(\alpha)), & \pi_0(p) = \alpha \\ (\lambda(p),\ 0,\ \sharp), & \text{otherwise} \end{cases}, p \in P$
$\mathbb{A}$: set of k asynchronous agents. $\pi_0 :\ \mathbb{A} \to V$ agents' initial placement $\pi :\ \mathbb{A} \to V$ location of agents.	k tokens π_0 : initial placement of tokens π : location of tokens.
Mobile Agent algorithm $\mathcal{A}$:	**Simulation message Passing algorithm $\mathcal{D}$:**
$Q_{\mathbb{A}}$: set of possible states of $\alpha \in \mathbb{A}$ $Q_{\mathbb{P}}$: set of possible states of $p \in \mathbb{P}$ $I_{\mathbb{A}} \subseteq Q_{\mathbb{A}}$: set of initial states of $\alpha \in \mathbb{A}$ $I_{\mathbb{P}} \subseteq Q_{\mathbb{P}}$: set of initial states of $p \in \mathbb{P}$	$\mathcal{M} = Q_{\mathbb{A}}$: set of possible messages $Q = Q_{\mathbb{P}} \times \{0,1\} \times Q_{\mathbb{A}} \cup \{\sharp\}$: set of possible states of $p \in P$ $I = I_{\mathbb{P}} \times \{0,1\} \times I_{\mathbb{A}} \cup \{\sharp\}$: set of initial states of $p \in P$
$state^{\mathcal{A}}(p),\ p \in \mathbb{P}$ $state^{\mathcal{A}}(\alpha),\ \alpha \in \mathbb{A}$ $\mathbb{M}$: multiset of agents in transit	$state(p) = \begin{cases} (state^{\mathcal{A}}(p),\ 1,\ state^{\mathcal{A}}(\alpha)), & \pi(p) = \alpha \\ (state^{\mathcal{A}}(p),\ 0,\ \sharp), & \text{otherwise} \end{cases}$ $, p \in P$ $M = \mathbb{M}$: multiset of messages in transit
Events of the relation $\vdash_p^{\alpha}$ of $\mathcal{A}_\alpha$	**Events of the relation $\vdash_p$ of $\mathcal{D}$, $p \in P$**
departure event: $(s,\ q,\ 0) \vdash_p^{\alpha} (s',\ q',\ out)$	**send event:** $\big((q,1,s),\ 0,\ \perp\big) \vdash_p \big((q',0,\sharp),\ out,\ s'\big)$
arrival event: $(s,\ q,\ in) \vdash_p^{\alpha} (s',\ q',\ 0)$	**receive event:** $\big((q,0,\sharp),\ in,\ s\big) \vdash_p \big((q',1,s'),\ 0,\ \perp\big)$
α **remains at** p**:** $(s,\ q,\ 0) \vdash_p^{\alpha} (s',\ q',\ 0)$	**internal event:** $\big((q,1,s),\ 0,\ \perp\big) \vdash_p \big((q',1,s'),\ 0,\ \perp\big)$

Table 2.2: Complexity of the Simulation of the Mobile Agent Algorithm $\mathcal{A}$ by Message Passing Algorithm $\mathcal{D}$

Message Complexity	$MSG(\mathcal{D}, C) = \sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) = TotalMove(\mathcal{A}, \mathbb{S})$
Bit Complexity	$Bit(\mathcal{D}, C) = \sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$
Time Complexity	$Time(\mathcal{D}, C) = Time\big(\mathcal{A}, \mathbb{S}\big)$
Local Space Complexity	$LocMem(\mathcal{D}, C) = \max_{p \in \mathbb{P}}\{LocMemWB(\alpha, \mathcal{A}, \mathbb{S})\} + \max_{\alpha \in \mathbb{A}}\{LocMemAg(\alpha, \mathcal{A}, \mathbb{S})\}$
Total Space Complexity	$Mem(\mathcal{D}, C) = \sum_{p \in \mathbb{P}} LocMemWB(\alpha, \mathcal{A}, \mathbb{S}) + \sum_{\alpha \in \mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$

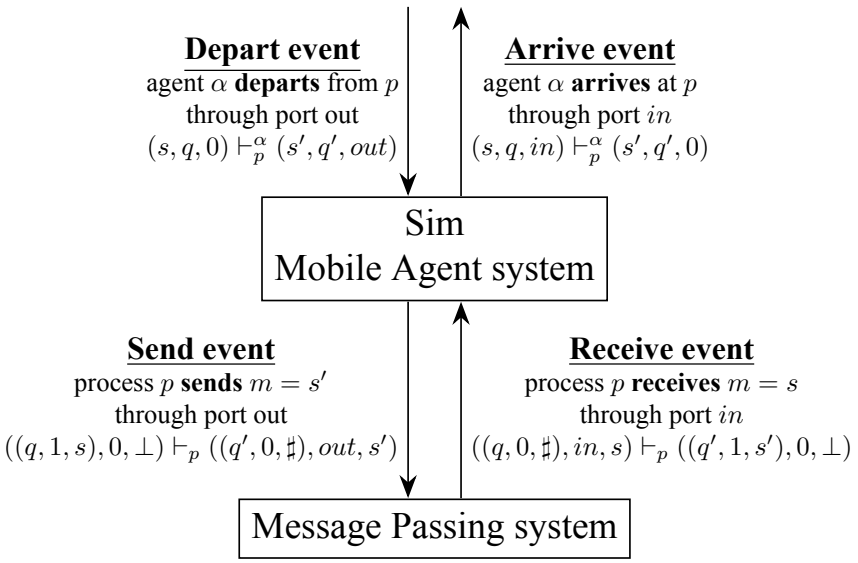

Figure 2.1: Simulation of Mobile agent algorithm by a Message Passing system, where the inputs of mobile agent system (depart events) are transformed into inputs of the message passing system (send events) and the outputs of the message passing system (receive events) are transformed into outputs of the mobile agent system (arrive events). Running the simulation algorithm on top of the message passing system produces the same appearance as does running the algorithm on top of the mobile agent system.

2.2 Simulation of synchronous MA Algorithm by a synchronous MP System

The simulation presented in section 2.1 can be modified appropriately in order to simulate synchronous mobile agent algorithms by synchronous message passing systems. Since in a synchronous mobile agent algorithm $\mathcal{A}$ in one round a mobile agent can execute a place and traverse a link, in the simulation algorithm $\mathcal{D}$ each token is received, processed and sent in one round by a process. Therefore, for every execution $\mathcal{E}_{\mathcal{D}}$ of the simulation algorithm $\mathcal{D}$ in the message passing system (P, C, λ') there exists an execution $\mathcal{E}_{\mathcal{A}}$ of mobile agent algorithm $\mathcal{A}$ in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$, where each state of each round of $\mathcal{E}_{\mathcal{D}}$ corresponds to a state in E_A in the same round. Thus, the **round complexity**(time) of the simulation algorithm $\mathcal{D}$ is the same as the round complexity(time) of algorithm $\mathcal{A}$ and the message complexity and space complexity of $\mathcal{D}$ is as presented in section 2.1

We note that the simulations that will be presented in chapter **??** of mobile agent algorithms with faults by message passing systems with faults, can be similarly modified and applied to the synchronous systems as well.

2.3 Simulation of a MP Algorithm by a MA System

In this section the mobile agent simulation algorithm of the message passing algorithm will not be described by an enumeration of the states and events, as in section 2.1, but by pseudo code [32]. The idea on simulating a message passing algorithm through a mobile agent algorithm is to make each agent responsible for executing the computational steps of a group of processes of the message passing algorithm.

Chalopin et al. proposed in [9] a procedure to simulate a message passing algorithm $\mathcal{D}$ on the system $(P, C, \lambda) = (V, E, \delta, \lambda)$ to a mobile agent algorithm $\mathcal{A}$, on the system $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda')$, where $\mathbb{S} = (V, E, \delta)$, there is one execution place $p \in \mathbb{P}$ on each vertex of V, $\mathbb{A}$ is a set of $k \geq 1$ asynchronous anonymous mobile agents, π_0 is any function $\mathbb{A} \to V$ and $\lambda'(p) = \lambda(p),\ p \in V$ and $\lambda'(\alpha) = \sharp,\ \forall \alpha \in \mathbb{A}$. In this procedure, even though agents and execution places might be anonymous, each agent $\alpha \in \mathbb{A}$ is able to compute a tree T_α, subgraph of G, by taking advantage of the edge labelling function δ_u. We call T_α the territory of agent α. Then each mobile agent $\alpha \in \mathbb{A}$ is responsible for simulating the execution of the local algorithm $\mathcal{D}_p$ for each of the corresponding process $p \in T_\alpha$ in his territory.

Partial Graph Traversal for the Construction of T_α

For the construction of T_α, each agent α starts from its homebase and begins a *partial graph traversal*. Since the system is anonymous, each agent stores in the variable (queue) $ePath$ the labels of the links traversed and when the agent wants to backtrack, he dequeues a link from $ePath$ and traverses that link. For each execution place u, on u's whiteboard there is a variable $visited$, which takes the value "yes", if at least one agent has visited u, otherwise takes the value "no". Initially, $visited_u \leftarrow$ "no", $\forall u \in V$. Similarly, for each execution place u, for each link $\delta_u(v)$, $v \in N_G(u)$ there is a variable traversed, where initially $traversed_u(\delta_u(v)) \leftarrow NULL,\ \forall u \in G\ \forall v \in N_G(u)$, which takes the value $traversed_u(\delta_u(v)) \leftarrow$ "T" if it has been explored and belongs to a spanning tree of an agent, otherwise if it is explored but does not belong to any tree $traversed_u(\delta_u(v)) \leftarrow$ "NT"

Each time agent α located at $u \in V(G)$ traverses an unexplored link $\{u, v\} \in E$, it marks it as explored, then visits vertex $v \in V(G)$ and adds $\delta_v(u)$ in ePath. If v is unvisited, α sets $visited_v \leftarrow$ "yes", $traversed_u(\delta_u(v)) \leftarrow$ "T" and $traversed_v(\delta_v(u)) \leftarrow$ "T", symbolizing that $\{u, v\} \in T_\alpha$. Then α chooses an unexplored edge $e = \{v, w\}$ such that $traversed_v(\delta_v(w)) = NULL$ to continue the graph traversal. If vertex u is visited, then agent α backtracks to v, by dequeuing the link $\delta_u(v)$ from $ePath$, and sets $traversed_u(\delta_u(v)) \leftarrow$ "NT" and $traversed_v(\delta_v(u)) \leftarrow$ "NT" , symbolizing that $\{u, v\} \notin T_\alpha$.

When there are no more unexplored links at a current node u, ($traversed_u(\delta_u(v)) \neq NULL,\ \forall v \in N_G(u)$), agent α backtracks, by dequeuing a link from $ePath$. When, agent α is located at $v \in V(G)$ and $ePath$ is empty, α knows that he is currently at his homebase. If α is at homebase and there are no more unexplored edges, the partial tree computation has ended, and by following the links labelled as T, agent α is able to traverse T_α and visit the vertices that correspond to processes for which he is responsible for simulating their actions in the message passing system.

Remarks: After agents finish the partial graph traversal:

- Every vertex of the graph G has been marked as visited by exactly one agent. Therefore, for every agent $\alpha,\ \beta \in \mathbb{A}$ it holds that $T_\alpha \cap T_\beta = \emptyset$
- Assuming that the graph of the navigation subsystem is connected, every vertex will be marked as visited by some agent.
- Consequently, every process in message passing model is assigned to exactly one agent in the mobile agent system, and each of the agents is responsible to execute the assigned processes computations.
- If $u_1 \in T_\alpha,\ u_2 \in T_\beta,\ \alpha, \beta \in \mathbb{A}$ then every (u_1, u_2)-path contains at lest one edge $e = \{w, z\}$ such that $traversed_w(\delta_w(z)) =$ "NT".

Message encoding

In the message passing algorithm, the events that can be performed by processes are to send a message via port j, receive a message via port j and internal events. The whiteboard of each vertex $whiteboard_u,\ u \in V$, has a variable $in - buf_u$, which is a FIFO queue containing all the received messages that haven't been read yet, and a variable TBD_u, which is a FIFO queue containing all the messages to be delivered to neighbours of u.

Send a message m via port j
Let $\{u, v\} \in G(E)$ be a link, such that $\delta_u(v) = j$, and suppose that in the local message passing algorithm $\mathcal{D}_u$ process u sends message m via port j. Suppose that $u \in T_\alpha$, for some agent $\alpha \in \mathbb{A}$. Then the action send message m via link j in the message passing system will be simulated in the mobile agent system as agent α dequeues message $< j, m >$ from TBD_u, traverses link j and writes in the $in - buf_v$ variable the tuple $< \delta_v(u), m >$ and backtracks through link $\delta_v(u)$. Agent α is able to write in the in-buf variable of vertex v, although vertex v doesn't necessarily belong to T_α.

Receive a message m from port j
Let $u \in G(V)$ and agent $\alpha \in \mathbb{A}$ located at u. The action receive a message m from

port j of the message passing algorithm is simulated in the mobile agent algorithm as dequeuing $< m, j >$ from the in-buf variable, and then continuing with the corresponding computations of algorithm $\mathcal{D}_u$ on process u.

Internal events:
For each internal event in the state of the process agents apply this events to the state of the corresponding execution place.

Procedure for simulating a message passing algorithm on a mobile agent setting

Algorithm 1 Simulation of a MP algorithm on a MA system [9]

Step 1: Each agent α constructs a tree T_α, which is result of partial graph traversal described before. This leads to a spanning forest of k trees, where each vertex $v \in V(G)$ belongs to exactly one T_α, for some agent α.

Step 2: Agent α executes the message passing algotihm $\mathcal{D}_v$ on the vertices $v \in T_\alpha$. Each time T_α is traversed by agent α, on every vertex $v \in T_\alpha$ $d \geq 1$ computational steps of $\mathcal{D}_v$ are executed.

Termination Detection

Although algorithm 1 of the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda')$ simulates algorithm $\mathcal{D}$ of the message passing system$(P, C, \lambda) = (V, E, \delta, \lambda)$, it does not inherit the termination property of algorithm $\mathcal{D}$. Therefore algorithm 1 should be modified so that if the simulated algorithm has the termination property, implicit or explicit, then the simulation algorithm has also the termination property, implicit or explicit respectively.

After each agent $\alpha \in \mathbb{A}$ traverses T_α and executes all computational steps of $\mathcal{D}_v$ at $v \in T_\alpha$, α returns at his homebase and becomes passive. In the event that a mobile agent $\beta \neq \alpha$, $\beta \in \mathbb{A}$ writes a message in the $in - buf$ variable of a vertex $u \in T_\alpha$, agent β should be able to wake up or notify agent α that there are more computations to execute on $u \in T_\alpha$.

For that reason, for each execution place $u \in T_\alpha$, $\forall \alpha \in \mathbb{A}$, the variable $visited_u$ is replaced by a variable $fatherlink_u$, that takes the values:

$$fatherlink_u = \begin{cases} NULL, \text{ if } u \text{ has not been visited yet} \\ -1, \text{ if } u \text{ is homebase of } \alpha \\ \delta_u(u, v), \text{ if } v \text{ is the father of } u \text{ in } T_\alpha \end{cases}$$

In this way, every agent β, that writes a message on $u \in T_\alpha$ is able to reach the homebase of α and wake up agent α. Additionally, every vertex that is homebase of an agent has a variable $fState_u$, which is the token "$Finished$" or the token "$NotFinished$", and indicates whether there are more computations to be executed on the vertices of T_α. Initially $fatherlink_u = NULL$, $\forall u \in V$ and $fState_{\pi_0(\alpha)} =$ "$NotFinished$", $\forall \alpha \in \mathbb{A}$.

Algorithm 2 Tree construction of T_α by mobile agent $\alpha \in \mathbb{A}$

Initially $fatherlink_u \leftarrow NULL$, $\forall u \in V$
$traversed_u(\delta_u(v)) \leftarrow NULL \ \forall u, v \ s.t. \ (u, v) \in E$

On wake up agent $\alpha \in \mathbb{A}$ is at homebase $\pi_0(\alpha) \in V$ and executes:
$fatherlink_{\pi_0(\alpha)} \leftarrow -1$
$ePath \leftarrow \emptyset$
$u \leftarrow \pi_0(\alpha)$
$\pi(\alpha) = \pi_0(\alpha)$
while $\exists v \in N_G(u)$ s.t. $traversed_u(\delta_u(v)) == NULL$ **or** $ePath \neq \emptyset$ **do**
  if $\exists v \in N_G(u)$ s.t. $traversed_u(\delta_u(v)) == NULL$ **then**
    $\pi(\alpha) \leftarrow v$ ▷ agent migrates to v
    $ePath.enqueue(\delta_v(u))$
    if $fatherlink_v = NULL$ **then**
      $fatherlink_v = \delta_v(u)$
      $traversed_v(\delta_v(u)) \leftarrow$ "T"
      $u \leftarrow v$
    else
      $traversed_v(\delta_v(u)) \leftarrow$ "NT"
      $(u, v) \leftarrow ePath.dequeue()$
      $\pi(\alpha) \leftarrow u$ ▷ agent backtracks to u
      $traversed_u(\delta_u(v)) \leftarrow$ "NT"
  else
    $(u, w) \leftarrow ePath.dequeue()$
    $\pi(\alpha) \leftarrow w$ ▷ agent backtracks to w
    $traversed_w(\delta_w(u)) \leftarrow$ "T"
    $u \leftarrow w$

The total number of agent moves of algorithm 2 is $2 \cdot E(G)$, since every edge of the graph G will be traversed by exactly one agent and exactly 2 times, once while exploring and once when backtracking.

The procedure that simulates a message passing algorithm $\mathcal{D}$ to a mobile agent algorithm $\mathcal{A}$ is described in algorithm 3.

Algorithm 3 Simulation of a terminating MP algorithm by an MA system [9]

Step 1: Each agent α executes the partial graph traversal of algorithm 2 and constructs a tree T_α. This leads to a spanning forest of k trees, where each vertex $v \in V(G)$ belongs to exactly one T_α, for some agent $\alpha \in \mathbb{A}$.
During that stage, when agent α backtracks from a vertex $u \in T_\alpha$, the $fatherlink_u$ of u is set to the dequeued value of $ePath$.

Step 2: Agent α executes the message passing algorithm $\mathcal{D}$ on the vertices of T_α.
Agent α begins a traversal of T_α if and only if the $fState$ of its homebase is "$NotFinished$", otherwise α becomes passive.
When α begins a traversal of T_α, α sets the $fState$ of its homebase to "$Finished$" and on every vertex $v \in T_\alpha$ executes $d \geq 1$ computational steps of $\mathcal{D}$.

If agent α delivers a message from vertex $u \in T_\alpha$ to vertex $v \in T_\beta$, then writes the message $< m, d_v(u) >$ in the $in - buf$ variable v and follows the $fatherLink$ of v to reach the homebase w of agent β.
If the $fState$ of w is "$Finished$", α sets $fState_w \leftarrow$ "$NotFinished$" and if β is "passive", agent α wakes him up and changes his state to "active".

Proposition 2.6. [9] Let $\mathcal{D}$ be a message passing algorithm implemented on the message passing system (P, C, λ), where $C = (V, E, \delta)$. Then algorithm 3 implemented on mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda')$, where $\mathbb{S} = (V, E, \delta)$ simulates algorithm $\mathcal{D}$.

Proof. By the construction of the simulation algorithm 3 we note that for every execution $\mathcal{E}$ of algorithm 3 in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda')$ there exists an equivalent execution $\mathcal{E}_\mathcal{D}$ of algorithm $\mathcal{D}$ in the message passing system (P, C, λ). □

Termination

Lemma 2.7. Let $\mathcal{D}$ be a message passing algorithm implemented on the message passing system (P, C, λ), where $C = (V, E, \delta)$. If algorithm $\mathcal{D}$ has the termination property then the simulation algorithm 3 implemented on the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda')$, where $\mathbb{S} = (V, E, \delta)$ has the termination property.

Proof. Algorithm 3 simulates algorithm $\mathcal{D}$, therefore for every execution $\mathcal{E}$ of algorithm 3 there exists an equivalent execution $\mathcal{E}_\mathcal{D}$ of $\mathcal{D}$.

If algorithm $\mathcal{D}$ terminates implicitly then all processes eventually become passive in $\mathcal{E}_\mathcal{D}$ and $\forall u \in P$ there is no applicable event. Hence, there exists a time in execution $\mathcal{E}$ such that all simulated events of algorithm $\mathcal{D}$ have been executed and there is no applicable simulated event for every $u \in \mathbb{P}$. By the construction of algorithm 3, each agent $\alpha \in \mathbb{A}$ is either passive or will eventually become passive after finishing the traversal of T_α and arriving at $homebase_\alpha$.

If algorithm $\mathcal{D}$ terminates explicitly then at least one process $u \in P$ detects termination in $\mathcal{E}_\mathcal{D}$; i.e. that all processes have their final values. By the construction of simulation algorithm 3, eventually some agent $\alpha \in \mathbb{A}$ located at $u \in \mathbb{P}$ detects termination ; i.e. that all whiteboards have their final values.

Hence the simulation algorithm 3 terminates explicitly. □

Complexity

Space Complexity

Proposition 2.8. Let $\mathcal{D} = (\vdash_p)_{p\in P}$ be a message passing algorithm implemented on the message passing system (P, C, λ), where $C = (V, E, \delta)$. Let $\mathcal{A} = (\vdash_p^\alpha)_{\alpha\in\mathbb{A},p\in\mathbb{P}}$ be the mobile agent algorithm 3 on the mobile agent system $(\mathbb{A}, \mathbb{P}), \mathbb{S}, \pi_0, \lambda'$, where $\mathbb{S} = C$. Then the **local space complexity** of agent $\alpha \in \mathbb{A}$ is:

$$LocMemAg(\alpha, \mathcal{A}, \mathbb{S}) = O\big(|V| \cdot \log\Delta + \max_{m\in\mathcal{M}} |m|\big)$$

where Δ is the maximum degree of the graph and m is the size/bits of messages $m \in \mathcal{M}$, and the **local space complexity** of execution place $p \in \mathbb{P}$ is:

$$LocMemWB(p, \mathcal{A}, \mathbb{S}) = O\big(\Delta\big) + Bit(\mathcal{D}, C) + LocMem(\mathcal{D}, C)$$

where $Bit(\mathcal{D}, C)$ is the total bits of messages exchanged $\mathcal{D}$ and $LocMem(\mathcal{D}, C)\big)$ is the local space complexity of the message passing algorithm.

Proof. When executing algorithm 3, each agent needs to have enough memory: to store the links traversed ($ePath$ variable) in order to be able to return to $homebase_\alpha$, to store the messages to be delivered (one message per delivery) when simulating the send event of process $u \in T_\alpha$ and to simulate the receive and internal events of process $u \in T_\alpha$. Therefore the **notebook** memory of agent α ($LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$) should be enough to:

- store $ePath \to O(n \cdot \log\Delta)$, where $\Delta = max_{u\in G(V)} deg(u)$
- store messages to be delivered: $O(\max_{m\in\mathcal{M}} |m|)$
- store $fStsate$ variable $\to O(1)$.
  (whether the agent is active or passive)

The **whiteboard** of each execution place $u \in \mathbb{P}$ ($LocMemWB(p, \mathcal{A}, \mathbb{S})$) corresponds to the memory of process u and should have enough memory to store:

- variable $visited_u \to O(1)$.
- variable $fatherLink_u \to O(\log\Delta)$, $\Delta = max_{u\in G(V)} deg(u)$
  (the link of the father of u in the tree T_α) .
- variable $traversed_u$: $O(\Delta)$,
- $in - buf_u$ variable $\to Bit(\mathcal{D}, C)$
  (the incoming messages: total number of bits that process u received in message passing algorithm $\mathcal{D}$)
- TBD_u variable $\to Bit(\mathcal{D}, C)$
  (the outgoing messages: total number of bits process u sent in message passing algorithm $\mathcal{D}$)
- the states of process $u \in P$ in $\mathcal{D} \to LocMem(\mathcal{D}, C)$

□

Move Complexity

The number of agent moves required for the simulation depends on:

- the number of messages exchanged in the message passing algorithm
- the number of agents, the graph topology, and therefore the size and depth of each tree T_α constructed by agent α
- the number of computational steps executed on each vertex $u \in T_\alpha$ on each traversal of T_α, by agent α, which depends on the message passing algorithm $\mathcal{D}$.

Depth of the tree:
Assume that the mobile agent system has k agents $\alpha_1, \cdots, \alpha_k$ and each agent α_i performs t_{α_i} traversals of T_{α_i} and then terminates.

If agent α_i simulates the action send a message from vertex $u \in T_{\alpha_i}$ to vertex $v \in T_{\beta_j}$, where j is not necessarily different from i, then the number of steps required by agent α_i is:

- 2 steps, one to arrive to vertex v and one to backtrack to u, and continue computations on u or move to the next vertex of T_{α_i}
- In the case of simulation with termination (algorithm 3), agent α_i needs to reach homebase of agent α_j and then backtrack to vertex v, which requires at most $2 \cdot depth(T_{\alpha_j})$ moves.

Remark: If agent α simulates the event send message m from u to v, if edge $\{u, v\}$ is marked as T , then agent α knows that $v \in T_\alpha$, and can omit going to its homebase, by having an extra local variable $fState_\alpha \leftarrow$ "NotFinished". If edge $\{u, v\}$ is marked as NT, then agent α doesn't know that $v \in T_\alpha$, and has to follow fatherlink to reach its homebase.

Therefore, if the total number of messages exchanged in the message passing algorithm $\mathcal{D}$ is $MSG(\mathcal{D}, C)$, then the total number of moves of agents to simulate the actions of sending messages is at most:

$$Move(\mathcal{A}, \mathbb{S}) = \Big(MSG(\mathcal{D}, C) \cdot (2 + 2 \cdot \max_{i \in \{1,\ldots,k\}} \{depth(T_{\alpha_i})\}) \Big)$$

Size of the tree
On step 1 of algorithms 1 and 3 every agent α_i traverses T_{α_i} t_{α_i} times and then terminates. For every traversal of T_{α_i}, α_i needs $2 \cdot |V(T_{\alpha_i})|$ moves to reach its homebase. Therefore the total number of moves performed by all the agents is at most:

$$\underbrace{2 \cdot |E|}_{\text{construction of } T_{\alpha_i}} + \underbrace{2 \cdot \sum_{i=1}^{k} t_{\alpha_i} \cdot |V(T_{\alpha_i})|}_{\text{traversal of } T_{\alpha_i}} + \underbrace{MSG(\mathcal{D}, C) \cdot (2 + 2 \cdot \max_{i \in \{1,\ldots,k\}} \{depth(T_{\alpha_i})\})}_{\text{message delivery}} =$$

$$= O\big(|E| + MSG(\mathcal{D}, C) \cdot |V| \big)$$

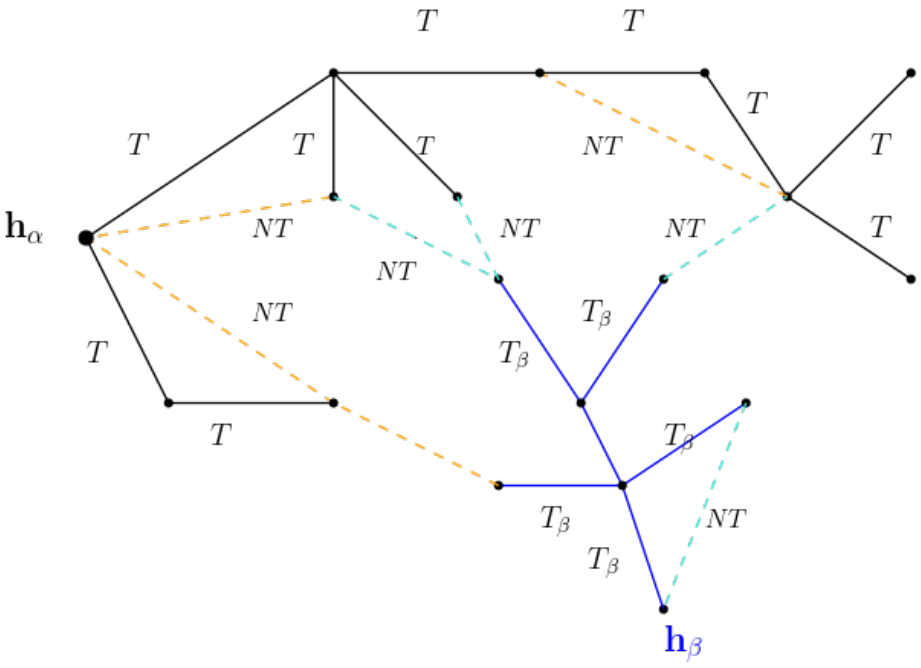

Figure 2.2: Example of tree construction of agents $\alpha, \beta \in \mathbb{A}$

Proposition 2.9. Let $\mathcal{D} = (\vdash_p)_{p \in P}$ be a message passing algorithm implemented on the message passing system (P, C, λ), where $C = (V, E, \delta)$. Let $\mathcal{A} = (\vdash_p^\alpha)_{\alpha \in \mathbb{A}, p \in \mathbb{P}}$ be the mobile agent algorithm 3 on the mobile agent system $(\mathbb{A}, \mathbb{P}), \mathbb{S}, \pi_0, \lambda'$, where $\mathbb{S} = C$. Then the **move complexity** of algorithm $\mathcal{A}$ is:

$$TotalMove(\mathcal{A}, \mathbb{S}) = \sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) = O\big(|E| + MSG(\mathcal{D}, C) \cdot |V|\big)$$

where $MSG(\mathcal{D}, C)$ is the message complexity of algorithm $\mathcal{D}$ on communication system C.

Remark: The simulation of message passing algorithm to mobile agent algorithm described in algorithm 2 is general and applicable to every message passing algorithm. However it is not always the most efficient in time complexity and in the total number of agents' moves.

For specific message passing algorithms more efficient simulations can be designed to reduce the number of agents' moves. For example Suzuki et al. in [31] present for the problem of gossiping a simulation of the message passing algorithm to a mobile algorithm that terminates within $2 \cdot MSG(\mathcal{D}, C)$ moves of agents, where $MSG(\mathcal{D}, C)$ is the total number of messages exchanged in the message passing algorithm.

2.4 Simulation of synchronous MP Algorithm by a synchronous MA System

The simulation presented in chapter 2.3 can be modified so that it can simulate a synchronous message passing algorithm by a synchronous mobile agent system. In the modified simulation algorithm one round of the message passing algorithm $\mathcal{D}$ will be simulated in multiple rounds of the simulation algorithm $\mathcal{A}$. More precisely, in each synchronous round r of algorithm $\mathcal{D}$ each agent $\alpha \in \mathbf{A}$ will traverse one time his tree α. When α visits each node $u \in T_\alpha$:

- It will collect and read all the messages that u received in the corresponding previous round (round $r-1$ of algorithm $\mathcal{D}$). If u hasn't received a messages of round $r-1$ from each one of its neighbors, then agent α will wait until every neighbor of u sends a message for round $r-1$.
- Then it execute the local algorithm $\mathcal{A}_u$, writes the messages to be sent in the TBD_u variable and delivers the messages of round r in the neighbors of u, as described in section 2.3. If in the local algorithm $\mathcal{A}_u$, u doesn't send a message to a process $v \in N_G(u)$, then agent α delivers a null message to v. Then agent α repeats this procedure at the next node of his tree T_α.

We note that the time that agent $\alpha \in \mathbf{A}$ will have to wait at node u at round $r-1$ to receive all the messages from round $r-1$ is finite since if agent is waiting for a message from neighbor v from round $r-1$ then:

- If $v \in T_\alpha$, then the message must have already been delivered, since α has already delivered all messages of nodes of T_α for round $r-1$.
- If $v \in T_\beta$, $\beta \neq \alpha \in \mathbf{A}$, then agent β is executing round $r-1$ of algorithm $\mathcal{D}$, since otherwise this would imply that agent α could not be executing round r. Therefore, in finite time agent β will deliver the messages of round $r-1$ of the local algorithm $\mathcal{D}_{\sqsubseteq}$.

Complexity

Time/Round Complexity

Let $H_\alpha = T_\alpha \cup \{N_G(u) : u \in T_\alpha\}$ be the subgraph of G containing the tree T_α and the neighbors of the vertices of T_α and let $E(T_\alpha)$ and $E(H_\alpha)$ denote the set of edges of T_α and H_α respectively. For every round of algorithm $\mathcal{D}$ simulation algorithm $\mathcal{A}$ requires:

- $2 \cdot \max_{\alpha \in \mathbf{A}} |T_\alpha|$ agent rounds for the partial tree traversal of T_α and the delivery of the messages in T_α
- For the message delivery:
  - $\forall u \in T_\alpha$, the messages with recipient u can be delivered to u during the T_α traversal.
  - $\forall u \in |H_\alpha \setminus T_\alpha|$ the messages with recipient u require $|E(H_\alpha) \setminus E(T_\alpha)|$ additional rounds for the message delivery.

Therefore if the round complexity of algorithm $\mathcal{D}$ is R then the worst case round complexity of the simulation algorithm $\mathcal{A}$ is:

$$Time(\mathcal{A}, \mathbb{S}) = Time(\mathcal{D}, C) \cdot 2 \cdot \max_{\alpha \in \mathbf{A}} |T_\alpha| + |E(H_\alpha) \setminus E(T_\alpha)|$$

Move Complexity of agents

The total moves performed by the agents on the execution of $\mathcal{A}$ is:

$$TotalMove(\mathcal{A}, \mathbb{S}) = \sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) = Time(\mathcal{D}, C) \cdot \big(\sum_{\alpha \in \mathbb{A}} (2 \cdot |T_\alpha| + 2 \cdot |E(H_\alpha) \setminus E(T_\alpha)|) \big)$$

$$= Time(\mathcal{D}, C) \cdot (2 \cdot n + 4 \cdot |E(G) \setminus \{ \bigcup_{\alpha \in \mathbb{A}} E(T_\alpha)|\})$$

where $Time(\mathcal{D}, C)$ is the time/round complexity of algorithm $\mathcal{D}$ on communication system C. Therefore the total moves performed by the agents are:

$$TotalMove(\mathcal{A}, C) = O\big(Time(\mathcal{D}, C) \cdot |V| + |E|\big)$$

CHAPTER 3

SIMULATIONS AMONG ADVERSARIAL MA AND MP MODELS

3.1 Simulation of a MP Algorithm by MA when agents might crash

A typical and realistic type of fault that can occur in the mobile agent model are crash failures of mobile agents, i.e. when some mobile agents crash and stop executing the mobile agent algorithm. Das et al. [13] proved that a message passing algorithm can be simulated in the mobile agent system even when agents might crash unexpectedly while traversing an edge, as long as at least one agent survives. They provided two algorithms in their work, one for the non anonymous setting and one for the anonymous setting, where neither agents nor execution places have unique identifiers. Their proposed algorithm for the anonymous setting, $AnSimulate$ [13], simulates a message passing algorithm in an anonymous mobile agent system with k mobile agents and at most $k-1$ crash failures.

In the setup phase of algorithm 4, agents run a partial graph traversal algorithm to obtain their territories, similarly to algorithm 2. Then, in algorithm 4 $AnSimulate$, each agent simulates the message passing algorithm on his territory and at the same time checks neighboring territories to see if there are any dead agents, by checking the delay of delivering the messages. If in a neighboring territory there is a slow or dead agent, the agent annexes the territory to his territory and continues simulating the message passing algorithm at the expanded territory.

The partition of the territories is self-balancing and even though some agents might crash, the algorithm ensures that the simulation proceeds correctly on every execution place and the work is equally distributed among the surviving agents.

Algorithm 4 AnSimulate [13]

Phase 0:
Each agent α executes algorithm 2 to construct partial spanning tree T_α, which is the agent's territory.This leads to a spanning forest of k trees, where each vertex $v \in V(G)$ belongs to exactly one T_α, for some agent α.
Let n_α be the size of T_α.
Agent α traverses T_α and executes algorithm $\mathcal{D}_p$ on $p \in T_\alpha$. α updates $state_p$ and if message m needs to be sent through port j, α appends $(m.j)$ to TBD queue.

for Phase $i \geq 1$: **do**

Step 1: Agent $\alpha \in \mathbb{A}$ (if alive) does a dfs traversal on tree T_α and on each execution place $p \in T_\alpha$ it visits:

if $TBD_p \neq \emptyset$ **then**
α delivers the messages of TBD_p

if $state_p =$ "$processing$" **then**
α continues with the local computations of algorithm $\mathcal{D}_p$

if $in-buf \neq \emptyset$ **then**
α receives the messages from $in-buf$

if $TBD = \emptyset$ and $in-buf = \emptyset$ **then**
$state_p \leftarrow' TERM'$
$fState_p = Finished$

Write $DONE(i, n_\alpha)$ on $whiteboard_p$

if $ANNEXED_{\pi_0(\alpha)} = (j, n_\beta, pathto(e'))$[1] **then**
α traverses $pathto(e')$ to arrive to node u and sets $traversed_u(e') \leftarrow$ "T"
$n_\alpha \leftarrow n_\alpha + n_\beta$
$T_\alpha \leftarrow T_\alpha + \{e'\} + T_\beta$
$ANNEXED_{\pi_0(\alpha)} = NULL$
Go to step 3

Step 2: Agent $\alpha \in \mathbb{A}$ starts a dfs traversal of its territory T_α. During the traversal for each external edge (i.e. $e \in E$ such that $traversed_u(e) = NT$ for some $u \in T_\alpha$) it traverses the edge $e = \{u, v\}$, and reads $DONE(j, n_\beta)$

if $j < i$ or ($j = i$ and $n_\beta < n_\alpha$) **then**
Go to homebase of $\beta \in \mathbb{A}$ (by following fatherlinks)
if $ANNEXED_{homebase_\beta} = NULL$ **then**
$ANNEXED_{homebase_\beta} \leftarrow (i, n_\alpha, pathto(e))$
$n_\alpha \leftarrow n_\alpha + n_\beta$
$T_\alpha \leftarrow T_\alpha + \{e\} + T_\beta$

Step 3: Agent $\alpha \in \mathbb{A}$ updates territory T_α to include all territories it annexed and those annexed by the agents it defeated.
The fatherlink of each node in T_α is updated and n_α is modified accordingly.
if $fState_u =$ "$Finished$" $\forall u \in T_\alpha$ **then**
Execute termination detection algorithm 5
else
Go to phase $i + 1$

[1] $pathto(e')$ is the path that contains edge labels from $\pi_0(\alpha)$ to e'

Algorithm 5 Termination Detection [13]

for r=1 to k do **do**
  if $\exists u \in T_\alpha$ such that $fState \neq$ "$Finished$" **then**
    return false
  else
    $fState_u \leftarrow$ "$Finished(r)$", $\forall u \in T_\alpha$
  for each $e = \{u, v\} \in E$ such that $u \in T_\alpha$, $traversed_u(\delta_u(v)) =$ "NT" **do**
    traverse e to arrive at $v \in T_\beta$, $\beta \in \mathbb{A}$
    if $fState_v \neq$ "$Finished$" **then**
      return false
    else if $fState =$ "$Finished(j)$" and $j < r$ **then**
      go to $homebase_\beta$
      if $fState_{homebase_\beta} \neq$ "$Finished(r)$" **then**
        $fState_{homebase_\beta} \leftarrow$ "$Finished(r)$"
        merge T_α and T_β
if $r = k$ **then**
  return true

Proposition 3.1. [13] Let $\mathcal{D}$ be a message passing algorithm implemented on the message passing system (P, C, λ), where $C = (V, E, \delta)$. Then algorithm AnSimulate 4 implemented on mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda')$, where $\mathbb{S} = (V, E, \delta)$ and at most $|\mathbb{A}| - 1 = k - 1$ agents crash, simulates algorithm $\mathcal{D}$.

Proof. By the construction of the simulation algorithm 4 we note that for every execution $\mathcal{E}$ of algorithm 4 in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda')$ where at most $k-1$ agents crash:

- $\forall u \in V\ \exists \alpha \in \mathbb{A}$ such that $u \in T_\alpha$ in phase 0.
- $\forall u \in V\ \exists \alpha \in \mathbb{A}$ such that $u \in T_\alpha$ in phase $i > 0$, where $\mathbb{A}$ is the set of alive agents.
- at least one agent survives and $\bigcup_{\alpha \in \mathbb{A}} T_\alpha = V$, where $\mathbb{A}$ is the set of alive agents.

Therefore, exists an equivalent execution $\mathcal{E}_\mathcal{D}$ of algorithm $\mathcal{D}$ in the message passing system (P, C, λ), since the algorithm 4 ensures that at least one agent will survive and correctly continue the simulation. □

Termination

Lemma 3.2. [13] Let $\mathcal{D}$ be a message passing algorithm implemented on the message passing system (P, C, λ), where $C = (V, E, \delta)$. If algorithm $\mathcal{D}$ has the termination property then the simulation algorithm 4 implemented on the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda')$, where $\mathbb{S} = (V, E, \delta)$ and at most $k - 1$ agents crash, terminates explicitly.

Proof. Algorithm 4 simulates algorithm $\mathcal{D}$, therefore for every execution $\mathcal{E}$ of algorithm 4 there exists an equivalent execution $\mathcal{E}_\mathcal{D}$ of $\mathcal{D}$.

If algorithm $\mathcal{D}$ terminates implicitly or explicitly then all processes eventually become passive in $\mathcal{E}_\mathcal{D}$ and $\forall u \in P$ there is no applicable event. Hence, there exists a time in execution $\mathcal{E}$ such that all simulated events of algorithm $\mathcal{D}$ have been executed

and there is no applicable simulated event for every $u \in \mathbb{P}$. Therefore the termination detection algorithm 5 will return true, since every message has been delivered and there are no messages in transit.

Hence the simulation algorithm 4 terminates explicitly. □

Complexity

Space Complexity

Proposition 3.3. Let $\mathcal{D} = (\vdash_p)_{p \in P}$ be a message passing algorithm implemented on the message passing system (P, C, λ), where $C = (V, E, \delta)$. Let $\mathcal{A}$ be the simulation mobile agent algorithm 4 and $\mathbb{S} = C$. Then the **local space complexity** of each agent is:

$$LocMemAg(\alpha, \mathcal{A}, \mathbb{S}) = O\big(|V| \cdot \log \Delta + \max_{m \in \mathcal{M}} |m|\big)$$

where $\Delta = \max_{u \in V} deg(u)$ and $|m|$ is the size in bits of message $m \in \mathcal{M}$ and the **local space complexity** of each execution place is:

$$LocMemWB(p, \mathcal{A}, \mathbb{S}) = O\big((\Delta) + MSG(\mathcal{D}, C) + LocMem(\mathcal{D}, C)\big)$$

where $MSG(\mathcal{D}, C)$ is the message complexity and $LocMem(\mathcal{D}, C)$ is the local space complexity of algorithm $\mathcal{D}$ implemented on communication system C.

Proof. The local memory requirements of each agent and execution place is similar with the proof of proposition 2.8 □

Move Complexity

Proposition 3.4. Let $\mathcal{D} = (\vdash_p)_{p \in P}$ be a message passing algorithm implemented on the message passing system (P, C, λ), where $C = (V, E, \delta)$. Let $\mathcal{A}$ be the simulation mobile agent algorithm 4 and $\mathbb{S} = C$. Then the **move complexity** of each agent $\alpha \in \mathbb{A}$ is:

$$Move(\alpha, \mathcal{A}, \mathbb{S}) = O\big((|E| + |V| \cdot k) \cdot MSG(\mathcal{D}, C)\big)$$

. Therefore the move complexity is for all the messages exchanged in algorithm $\mathcal{D}$ is $O\big((|E| + |V| \cdot k) \cdot MSG(\mathcal{D}, C)\big)$, where $MSG(\mathcal{D}, C)$ is the message complexity of algorithm $\mathcal{D}$.

Proof. The complexity overhead of algorithm 4 $AnSimulate$ is $O((|E| + |V|k))$ for each message exchanged in algorithm $\mathcal{D}$, where $|E|$ is the number of edges, $|V|$ is the number of nodes in the graph, k is the initial number of mobile agents and $MSG(\mathcal{D}, C)$ is the total number of messages exchanged in the message passing algorithm $\mathcal{D}$. □

Fault Tolerant simulation of Message Passing Algorithms in the non Anonymous setting

In the non anonymous case, where agents and execution places have unique identifiers, the authors of [13] proposed the fault-tolerant algorithm $DisSimulate$, to simulate any message passing algorithm by a mobile agent system, which is more efficient in terms of agents moves. The overall agent moves of $DisSimulate$ is $O((|E| + |V| \cdot MSG(\mathcal{D}, C)) \cdot k)$, where $|V|$ is the number of processes/execution places, $|E|$ is the number of links, $k = |\mathbb{A}|$ is the total number of agents and $MSG(\mathcal{D}, C)$ is the number of

messages exchanged in the message passing algorithm $\mathcal{D}$ on the communication system C.

This was later improved by Gotoh et. al. in [19], where they proposed a fault-tolerant algorithm for the non-anonymous case that simulates any message passing algorithm with $O((|E| + MSG(\mathcal{D}, C)) \cdot f)$ total agent moves, where m is the number of links, $MSG(\mathcal{D}, C)$ is the number of messages exchanged in the message passing algorithm and $f \leq k - 1$ is an upper bound on the number of crashed agents.

3.2 Always Dead Processes in MP (ADP-MP) and Black holes in MA (BH-MA)

3.2.1 MP Model with Always dead Processes (ADP)

In the message passing model an Always dead process (ADP) (or initially dead process [17] or initial failures) is a type of benign failure. An always dead process is a process that does not execute a single step of the algorithm [32]. A message passing model with always dead processes is defined by $(P,\ C,\ D,\ \lambda)$ where:

- $P,\ C,\ \lambda$ are as defined in section 1.2
- $D \subsetneq P$ is a set of always dead processes or faulty processes.
- $P \setminus D$ is the set of correct or non faulty processes.

Message Passing algorithm with always dead processes

A message passing algorithm with always dead processes is defined as in section 1.2 with the modifications that $\forall d \in D$, d does not execute any event, hence the local algorithm $\mathcal{D}_d$ will have no events and the $state(d)$ will remain the initial state $\lambda(d)$. Without loss of generality, we can assume that every message m that is sent to and ADP is ignored and is removed from the list of messages in transmission. This is modelled as an **ignore event**:

$$(\lambda(d),\ in,\ m) \vdash_d (\lambda(d),\ 0,\ \perp)$$

where $\lambda(d)$ is the initial state of process d, in is the port through which m is received. $M \leftarrow M \setminus \{p, m, d\}$ where $p \in P$ is such that $\delta_d(p) = in$

The termination property is modified to depend only to correct processes, regardless of the faulty behavior of the always dead processes.

The interaction of an always dead process with the message passing communication system is depicted in figure 3.1. Every message that arrives to the always dead process from the communication system is ignored. Therefore without loss of generality, the interface of the message passing system contains also the ignore events of the always dead processes, as depicted in figure 3.2

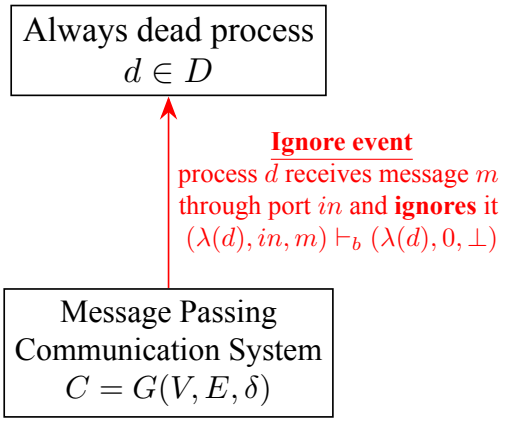

Figure 3.1: The interface of an Always Dead process with the message passing communication system. When a message m is received by $d \in D$, it is ignored and the state of the process d doesn't change.

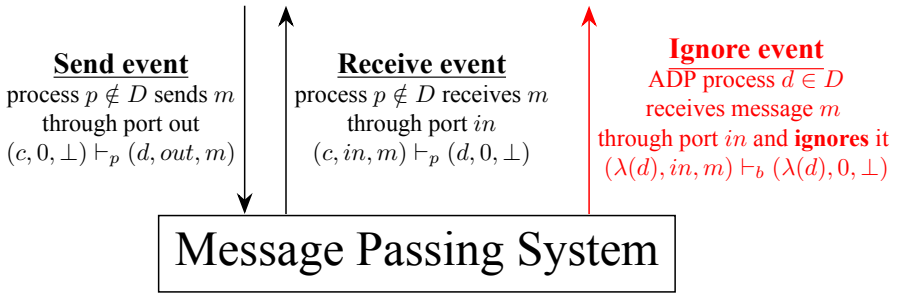

Figure 3.2: The interface of a Message passing system with always dead process. The inputs of the message passing system are the send events and the outputs are the receive events and the ignore events.

3.2.2 MA Model with Black holes (BH)

In the mobile agent model a *black hole* (BH) is a type of stationary host attack. More precisely, a *black hole* is a stationary process located at an execution place $b \in \mathbb{P}$ that destroys every agent that visits b, without leaving any observable trace to the surviving agents.

A mobile agent system with a set of black holes is defined as $(\mathbb{A},\ \mathbb{P},\ \mathbb{B},\ \mathbb{S},\ \pi_0,\ \lambda)$ where:

- $\mathbb{A},\ \mathbb{P},\ \mathbb{S}, \lambda$ are defined as in section 1.3.
- $\mathbb{B} \subsetneq \mathbb{P}$ is a set of black holes that destroy any visiting agent without leaving any trace.
- $\mathbb{P} \setminus \mathbb{B}$ is the set of safe execution places
- $\pi_0 : \mathbb{A} \to \mathbb{P} \setminus \mathbb{B}$ agents are initially located on safe nodes/execution places.

Mobile Agent Algorithm with black holes

A mobile agent algorithm with a set of black holes is defined as in section 1.3 with the following modifications:

- The set $Q_{\mathbb{A}}$ of possible states of the agents contains a special terminal state s_{BH} that indicates that an agent died when visiting a black hole $b \in \mathbb{B}$
- When agent $\alpha \in \mathbb{A}$ visits a black hole $b \in \mathbb{B}$ the only applicable event is the **destroy event**:

  $$(s, q_{BH}, in) \vdash_b^{\alpha} (s_{BH}, q_{BH}, 0)$$

  where s is the old state of agent $\alpha \in \mathbb{A}$, q is the old state of $b \in \mathbb{B}$, in is the port through which α arrived to b, s_{BH} is the new and terminal state of α and q' is the new state of $b \in \mathbb{B}$ and there is no other applicable event that involves agent $\alpha \in \mathbb{A}$
- after a destroy event of agent $\alpha \in \mathbb{A}$ on black hole $b \in \mathbb{B}$:
  $\mathbb{M} \leftarrow \mathbb{M} \setminus \{(p', \alpha, b)\}$ where $p' \in \mathbb{P}$ is such that $\delta_p(p') = in$
  $\pi(\alpha) \leftarrow \perp$
  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{\alpha\}$
- The termination property is modified to depend only to the surviving agents, i.e. agents that are not destroyed by the black hole.

The interaction of a black hole $b \in \mathbb{B}$ with the mobile agent communication system is depicted in figure 3.3. Every agent that arrives to the black hole from the navigation system is destroyed. Therefore without loss of generality, the interface of the message passing system contains also the destroy events of the black hole, as depicted in figure 3.4

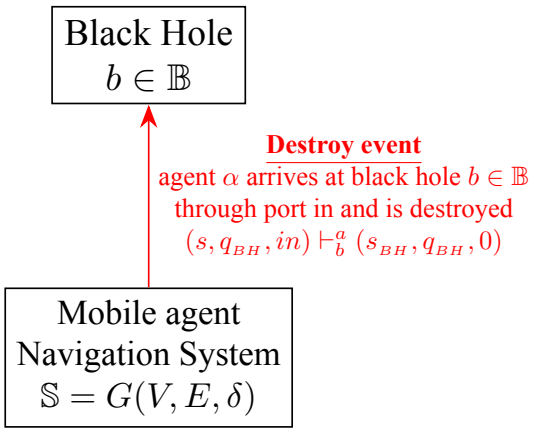

Figure 3.3: The interface of a Black Hole with the mobile agent navigation system. When an agent $\alpha \in \mathbb{A}$ arrives at $b \in \mathbb{B}$, it is destroyed, without leaving any trace.

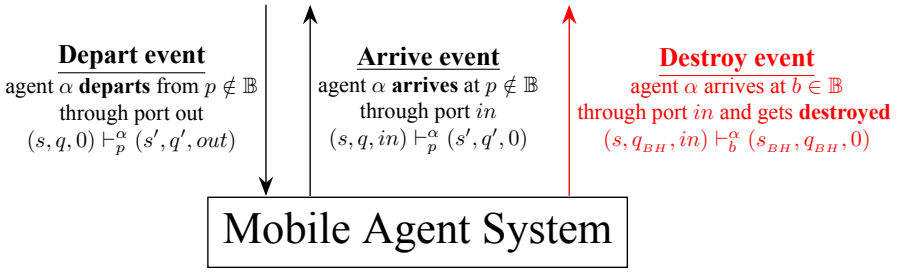

Figure 3.4: The interface of a Mobile agent system with black holes. The inputs of the mobile agent navigation system are the depart events and the outputs are the arrive events and the destroy events.

3.3 Simulations among BH-MA and ADP-MP model

3.3.1 Simulation of a BH-MA algorithm by an ADP-MP algorithm

A Mobile agent algorithm, resilient to a set of black holes $|\mathbb{B}|$, where at most $k-1 = |\mathbb{A}| - 1$ agents are destroyed, can be simulated by a message passing system with a set D of at most $|\mathbb{B}|$ always dead processes. The idea of the simulation is similar with section 2.1. In the case that in the message passing system a process $p \in P \setminus D$ sends a message/token to an always dead process, the message is ignored but the simulation algorithm remains resilient, since this corresponds to the arrival and destruction of an agent to a black hole in the mobile agent algorithm. Therefore, since in the mobile agent algorithm at most $k-1$ agents are destroyed by the black holes, then in the simulation algorithm, the simulation will proceed correctly despite the loss of the tokens sent to always dead processes.

Furthermore, it is possible to simulate black holes by always dead processes. When an agent arrives at a black hole, the only applicable event is the destroy event, where the agent is destroyed, so in the simulation by an always dead process every incoming message is ignored, and the state of the always dead process does not change.

More formally, let $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$ be a mobile agent system with a set of black holes $\mathbb{B}$, where $\mathbb{S} = (V, E, \delta)$ is the navigation subsystem and $\mathcal{A} = (\vdash_p^\alpha)_{\alpha \in \mathbb{A},\, p \in \mathbb{P}}$. The mobile agent algorithm $\mathcal{A}$ can be simulated in the message passing system (P, C, D, λ') in algorithm $\mathcal{D}$ defined as in section 2.1 with the following modifications:

- Each non faulty process $p \in P \setminus D$ of the message passing system corresponds to a safe execution place $p \in \mathbb{P} \setminus \mathbb{B}$ of the mobile agent system and executes algorithm $\mathcal{D}_p$ as defined in section 2.1.
- Each always dead process $b \in D$ of the message passing system corresponds to a black hole $b \in \mathbb{B}$.
- If a token $t(\alpha)$, $\alpha \in \mathbb{A}$ is sent to an always dead process, then it is ignored.
- Without loss of generality we can assume that when a message/token is sent to an always dead process in the simulation algorithm $\mathcal{D}$ then it is removed from the set M of messages in transit.

Proposition 3.5. Let $\mathcal{A} = (\vdash_p^\alpha)_{a \in \mathbb{A},\, p \in \mathbb{P}}$ be a mobile agent algorithm implemented on the mobile agent system with black holes $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$. Let $\mathcal{D} = (\mathcal{D}_p)_{p \in P}$

be the message passing algorithm defined above on the message passing system with always dead processes $(P,\ C,\ D\ \lambda')$.
Then algorithm $\mathcal{D}$ of the message passing system (P, C, D, λ') simulates algorithm $\mathcal{A}$ of the mobile agent system with respect to the non faulty processes.

Proof. Let $\mathcal{E}_{\mathcal{D}}$ be an execution of algorithm $\mathcal{D}_p,\ p \in P$ in the message passing system $(P,\ C,\ D,\ \lambda')$, f the simulation relation and h the events' mapping described above. By the construction of algorithm $\mathcal{D}_p$ from algorithm $\mathcal{A}$ of the system $(\mathbb{A},\ \mathbb{P},\ \mathbb{B},\ \mathbb{S},\ \pi_0,\ \lambda)$ we note that:

1. $f(\lambda') \cap I_{\mathbb{A}} \neq \emptyset$
2. If $s = (state_i, M_i)$, for some $i \in \mathbb{N}$ is a reachable configuration of $\mathcal{D}$, $u \in f(s)$, $u = (state_j^{\mathcal{A}}, \mathbb{M}_j, \pi_j), \in \mathbb{N}$ is a reachable configuration of $\mathcal{A}$ and π is an applicable event on s that changes the state from s to $s' = (state_{i+1}, M_{i+1})$ then:

   - If π is a **send event** from $p \in P$, where $state_i(p) = (q, 1, r) \neq (q_{_{BH}}, 1, r)$ to $p' \in P$:
   $$((q, 1, r), 0, \perp) \vdash_p ((q', 0, \sharp), out, r')$$
   where $state_{i+1}(p) = (q', 0, \sharp)$ and $M_{i+1} = M_i \cup \{(p, r', p')\}$ then for the **depart event** π' of $\mathcal{A}$:
   $$(r, q, 0) \vdash_p^{\alpha} (r', q', out)$$
   on the execution place $p \in \mathbb{P}$ with $state_j^{\mathcal{A}}(p) = q \neq q_{_{BH}}$, $state_j^{\mathcal{A}}(\alpha) = r$, $\pi_j(\alpha) = p$ and $state_{j+1}^{\mathcal{A}}(p) = q'$, $state_{j+1}^{\mathcal{A}}(\alpha) = r'$, $\mathbb{M}_{j+1} = \mathbb{M}_j \cup \{(p, \alpha, p')\}$, $\pi_{j+1}(\alpha) = \perp$, we have that
   $$u' = (state_{j+1}^{\mathcal{A}}, \mathbb{M}_{j+1}, \pi_{j+1}) \in f(s')$$
   $$h(\pi') = \pi$$
   - If π is a **receive event** of $p \in P$, where $state_i(p) = (q, 0, \sharp) \neq (q_{_{BH}}, 0, \sharp)$ from $p' \in P$
   $$((q, 0, \sharp), in, r) \vdash_p ((q', 1, r'), 0, \perp)$$
   where $state_{i+1}(p) = (q', 1, r')$ and $M_{i+1} = M_i \setminus \{(p', r, p)\}$ then for the **arrive event** π' of $\mathcal{A}$:
   $$(r, q, in) \vdash_p^{\alpha} (r', q', 0)$$
   on the execution place $p \in \mathbb{P}$ with $state_j^{\mathcal{A}}(p) = q$, $state_j^{\mathcal{A}}(\alpha) = r$, $\pi_j(\alpha) = \perp$ and $state_{j+1}^{\mathcal{A}}(p) = q'$, $state_{j+1}^{\mathcal{A}}(\alpha) = r'$, $\mathbb{M}_{j+1} = \mathbb{M}_j \setminus \{(p', \alpha, p)\}$, $\pi_{j+1}(\alpha) = p$, we have that
   $$u' = (state_{j+1}^{\mathcal{A}}, \mathbb{M}_{j+1}, \pi_{j+1}) \in f(s')$$
   $$h(\pi') = \pi$$
   - If π is an **internal event** of $p \in P$,where $state_i(p) = (q, 1, r) \neq (q_{_{BH}}, 1, r)$:
   $$((q, 1, r), 0, \perp) \vdash_p ((q', 1, r'), 0, \perp)$$

where $state_{i+1}(p) = (q', 1, r')$ and $M_{i+1} = M_i$ then for the **internal event** π' of $\mathcal{A}$:

$$(r, q, 0) \vdash_p^{\alpha} (r', q', 0)$$

on the execution place $p \in \mathbb{P}$ with $state_j^{\mathcal{A}}(p) = q$, $state_j^{\mathcal{A}}(\alpha) = r$, $\pi_j(\alpha) = p$ and $state_{j+1}^{\mathcal{A}}(p) = q'$, $state_{j+1}^{\mathcal{A}}(\alpha) = r'$, $\mathbb{M}_{j+1} = \mathbb{M}_j$, $\pi_{j+1}(\alpha) = p$, we have that

$$u' = (state_{j+1}^{\mathcal{A}}, \mathbb{M}_{j+1}, \pi_{j+1}) \in f(s')$$

$$h(\pi') = \pi$$

- If π is an **ignore event** of $p \in D$ from $p' \in P$:

$$((q_{BH}, 0, \sharp), in, r) \vdash_p ((q_{BH}, 0, \sharp), 0, \perp)$$

where $state_i(p) = state_{i+1}(p) = (q_{BH}, 0, \sharp)$, and $M_{i+1} = M_i \backslash \{(p', r, p)\}$ then for the **destroy event** π' of $\mathcal{A}$:

$$(r, q, in) \vdash_p^{\alpha} (r', q', 0)$$

on the execution place $p \in \mathbb{P}$ with $state_j^{\mathcal{A}}(p) = state_{j+1}^{\mathcal{A}}(p) = q_{BH}$, $state_j^{\mathcal{A}}(\alpha) = r$, $\pi_j(\alpha) = \pi_{j+1}(\alpha) = \perp$ and , $\mathbb{M}_{j+1} = \mathbb{M}_j \setminus \{(p', \alpha, p)\}$, we have that

$$u' = (state_{j+1}^{\mathcal{A}}, \mathbb{M}_{j+1}, \pi_{j+1}) \in f(s')$$

$$h(\pi') = \pi$$

Therefore, since $|D| \leq |\mathbb{B}|$, for any applicable event π we have that there exists an execution segment $\mathcal{E}_{\mathcal{A}}$ of $\mathcal{A}$ such that: $h(trace(\mathcal{E}_{\mathcal{A}})) = \pi$

Hence, by definition 1.14 the message passing system $(P,\ C,\ D,\ \lambda')$ simulates the mobile agent system $(\mathbb{A},\ \mathbb{P},\ \mathbb{B},\ \mathbb{S},\ \pi_0,\ \lambda)$ □

Simulation of a BH by an ADP

In the simulation of a Mobile Agent Algorithm by a Message Passing System, it is possible to simulate the black holes by an always dead process. In this case the initial state of the always dead processes is $(q_{BH)},\ 0,\ \sharp)$ and the **destroy event** of agent $\alpha \in \mathbb{A}$ by a black hole $b \in \mathbb{B}$:

$$(s, q_{BH}, in) \vdash_b^{\alpha} (s_{BH}, q_{BH}, 0)$$

is mapped in the message passing system as an **ignore event** of $b \in D$ in algorithm $\mathcal{D}$:

$$((q_{BH},\ 0,\ \sharp),\ in,\ s) \vdash_b ((q_{BH}),\ 0,\ \sharp),\ 0,\ \perp)$$

We note that after the destroy event agent α terminates and won't be involved in any other event. In the simulation the destroy event is not translated as an event since token $t(\alpha)$ will be ignored by b and there will be no other event involving token $t(\alpha)$. The simulation of a black hole by an always dead process is depicted in figure 3.6

Termination

Lemma 3.6. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm implemented on the mobile agent system with blackholes ($\mathbb{A}$, $\mathbb{P}$, $\mathbb{B}$, $\mathbb{S}$, π_0, λ). Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system with always dead processes (P, C, $D\,\lambda'$). If algorithm $\mathcal{A}$ has the termination property then algorithm $\mathcal{D}$ has the termination property.

Proof. Algorithm $\mathcal{D}$ simulates algorithm $\mathcal{A}$ with respect to the faulty processes, therefore for every execution $\mathcal{E}_\mathcal{D}$ of $\mathcal{D}$ with a set of always dead processes D there exists an equivalent execution $\mathcal{E}_\mathcal{A}$ of $\mathcal{A}$ with a set of blackholes $\mathbb{B}$.

If algorithm $\mathcal{A}$ terminates implicitly then all agents that are not destroyed by the black holes eventually become passive in $\mathcal{E}_\mathcal{A}$ and $\forall\alpha \in \mathbb{A}$ located at $p \in \mathbb{P} \setminus \mathbb{B}$ there is no applicable event of the relation $\vdash_p^\alpha$. By the mapping of events in the simulation message passing algorithm $\mathcal{D}$ it is implied that $\forall p \in P \setminus D$ there is no applicable event of the relation $\vdash_p$ on the execution $\mathcal{E}_\mathcal{D}$. Hence $\mathcal{D}$ terminate implicitly.

If algorithm $\mathcal{A}$ terminates explicitly then at least one agent $\alpha \in \mathbb{A}$ located at $p \in \mathbb{P} \setminus \mathbb{B}$ detects termination in $\mathcal{E}_\mathcal{A}$; i.e. that all execution places have their final values. By the construction of $\mathcal{D}$, process $p \in P \setminus D$ with token $t(\alpha)$ detects termination ; i.e. that all processes have their final values. Hence $\mathcal{D}$ terminates explicitly. □

Remarks: The simulation of a mobile agent algorithm $\mathcal{A}$ of the mobile agent system to a message passing algorithm $\mathcal{D}$ of the message passing system as described above is similar to the simulation in section 2.1. By the construction of the simulation algorithm $\mathcal{D}$ there is a correspondence between the $whiteboard_p$, $p \in \mathbb{P}\setminus\mathbb{B}$ and the local memory of the corresponding process $p \in P \setminus D$, between $notebook_\alpha$, of mobile agent $\alpha \in \mathbb{A}$ and the messages exchanged in the message passing algorithm $\mathcal{D}$ and between the set of blackholes $\mathbb{B}$ and the set of always dead processes D.

A mobile agent algorithm $\mathcal{A}$ that tolerates t black holes has the requirement that at least one agent survives and completes the problem requirements. Therefore, in the simulation algorithm $\mathcal{D}$ processes will complete the problem requirements by sending at most $k-1$ messages to the faulty process.

Complexity

Message Complexity

Proposition 3.7. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm that tolerates t black holes implemented on the mobile agent system ($\mathbb{A}$, $\mathbb{P}$, $\mathbb{B}$, $\mathbb{S}$, π_0, λ), where $|\mathbb{B}| \leq t$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system with always dead processes (P, C, D, λ'), where $|D| \leq t$.
Then the **total number of messages exchanged** during the execution of $\mathcal{D}$ is:

$$MSG(\mathcal{D}, C) = \sum_{\alpha\in\mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) = TotalMove(\mathcal{A}, \mathbb{S})$$

where $Move(\alpha, \mathcal{A}, \mathbb{S})$ is the move complexity of agent $\alpha \in \mathbb{A}$ in the execution of $\mathcal{A}$.
The total number of messages sent to the set of always dead processes D is at most k-1, where $k = |\mathbb{A}|$.
The **total size of messages exchanged** is

$$Bit(\mathcal{D}, C) = \sum_{\alpha\in\mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha\mathcal{A}, \mathbb{S})$$

where $LocMemAg(\alpha\mathcal{A}, \mathbb{S})$ is the local space complexity of agent $\alpha \in \mathbb{A}$, i.e. the size of notebook of α.

Proof. The proof is similar with the proof of proposition 2.3 with the modification that if in the simulation algorithm $\mathcal{D}$ there is a send event to an always dead process $b \in D$, then in algorithm $\mathcal{A}$ agent $\alpha \in \mathbb{A}$ arrives to black hole $b \in \mathbb{B}$ and is destroyed. □

Time Complexity

Proposition 3.8. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm that tolerates t black holes implemented on the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{B},\, \mathbb{S},\, \pi_0,\, \lambda)$, where $|\mathbb{B}| \leq t$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system with always dead processes $(P,\, C,\, D,\, \lambda')$, where $|D| \leq t$ and $C = \mathbb{S}$. The **time complexity** of algorithm $\mathcal{D}$ is:

$$Time(\mathcal{D}, C) = Time(\mathcal{A}, \mathbb{S})$$

where $Time(\mathcal{A}, \mathbb{S})$ is the time complexity of mobile agent algorithm $\mathcal{A}$ on the navigation system $\mathbb{S}$.

Proof. The proof is similar with the proof of proposition 2.4 □

Space Complexity

Proposition 3.9. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm that tolerates t black holes implemented on the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{B},\, \mathbb{S},\, \pi_0,\, \lambda)$, where $|\mathbb{B}| \leq t$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system with always dead processes $(P,\, C,\, D,\, \lambda')$, where $|D| \leq t$ and $C = \mathbb{S}$. Then the **local space complexity** of algorithm $\mathcal{D}$ is:

$$LocMem(\mathcal{D}, C) = \max_{p\in\mathbb{P}} \{LocMemWB(p, \mathcal{A}, \mathbb{S})\} + \max_{\alpha\in\mathbb{A}} \{LocMemAg(\alpha, \mathcal{A}, \mathbb{S})\}$$

where $LocMemWB(p, \mathcal{A}, \mathbb{S})$ is the local space complexity of execution place $p \in \mathbb{P}$ and $LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$ is the local space complexity of agent $\alpha \in \mathbb{A}$ of algorithm $\mathcal{A}$ on the navigation system $\mathbb{S}$.
The **total space complexity** of algorithm $\mathcal{D}$ is

$$Mem(\mathcal{D}, C) = \sum_{p\in\mathbb{P}} LocMemAg(p, \mathcal{A}, \mathbb{S}) + \sum_{\alpha\in\mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$$

Proof. The proof is similar with the proof of proposition 2.5 □

The simulation is summarised in tables 3.1, 3.2 and figure 3.5, and the simulation of black holes by always dead processes is summarised in figure 3.6. The complexity of the simulation is summarised in table 3.3

Table 3.1: Simulation of Mobile Agent Algorithm $\mathcal{A}$ with black holes by a Message Passing Algorithm $\mathcal{D}$ with always dead processes

Mobile Agent Model with black holes	**Message Passing Model with always dead processes**
System: $(\mathbb{A},\ \mathbb{P},\ \mathbb{B},\ \mathbb{S},\ \pi_0,\ \lambda)$ $\mathbb{S} = (V,\ E,\ \delta)$ navigation subsystem $\mathbb{P}$ or V : execution places, equipped with a whiteboard E : migration ports $\delta_u : N_{_G}(u) \to [1,\ deg_G(u)] :\ u \in V$ port labelling function λ : initial states of places and agents $\mathbb{A}$: set of k asynchronous mobile agents. $\pi_0 :\ \mathbb{A} \to V$ initial placement of agents. $\pi :\ \mathbb{A} \to V$ location of agents.	**System:** $(P,\ C,\ D,\ \lambda')$ $C = (V,\ E,\ \delta)$ communication subsystem P or V : asynchronous processes E : communication channels $\delta_u : N_{_G}(u) \to [1,\ deg(u)] :\ u \in V$ port labelling function λ' : initial states of processes $\lambda'(p) = \begin{cases} (\lambda(p),\ 1,\ \lambda(\alpha)), & \text{if } \pi_0(p) = \alpha \\ (\lambda(p),\ 0,\ \sharp), & \text{otherwise} \end{cases}, p \in P$ k tokens π_0 : initial placement of tokens. π : location of tokens.
Black hole $b \in \mathbb{B} \subsetneq \mathbb{P}$**:** a stationary process that destroys any incoming agent $s_{_{BH}}$: the state of the destroyed agent	**Always dead process** $b \in D \subsetneq P$**:** a process that does not execute a single step of the algorithm. $state(b) = \lambda'(b)$ during the execution of $\mathcal{D}$
Mobile agent algorithm $\mathcal{A}$ $Q_{\mathbb{A}}$: set of possible states of $\alpha \in \mathbb{A}$ $Q_{\mathbb{P}}$: set of possible states of $p \in \mathbb{P}$ $I_{\mathbb{A}} \subseteq Q_{\mathbb{A}} \setminus \{s_{_{BH}}\}$: set of initial states of α $I_{\mathbb{P}} \subseteq Q_{\mathbb{P}}$: set of initial states of $p \in \mathbb{P}$ $state^{\mathcal{A}}(p),\ p \in \mathbb{P}$ $state^{\mathcal{A}}(\alpha),\ \alpha \in \mathbb{A}$ $\mathbb{M}$: multiset of agents in transit	**Simulation message passing algorithm** $\mathcal{A}$ $\mathcal{M} = Q_{\mathbb{A}} \setminus \{s_{_{BH}}\}$: set of possible messages $Q = Q_{\mathbb{P}} \times \{0,1\} \times Q_{\mathbb{A}} \cup \{\sharp\} \setminus \{s_{_{BH}}\}$: set of possible states of $p \in P$ $I = I_{\mathbb{P}} \times \{0,1\} \times I_{\mathbb{A}} \cup \{\sharp\}$: set of initial states of $p \in P$ $state(p) =$ $= \begin{cases} (state^{\mathcal{A}}(p),\ 1,\ state^{\mathcal{A}}(\alpha)), & \pi(p) = \alpha \\ (state^{\mathcal{A}}(p),\ 0,\ \sharp), & \text{otherwise} \end{cases}$ $p \in P \setminus D$ $M = \mathbb{M}$: multiset of messages in transit

Table 3.2: Simulation of Mobile Agent Algorithm $\mathcal{A}$ with black holes by a Message Passing Algorithm $\mathcal{D}$ with always dead processes

Mobile Agent Model with black holes	**Message Passing Model with always dead processes**
Events of the relation $\vdash_p^\alpha$ of $\mathcal{A}$ $h: events(\mathcal{A}) \to events(\mathcal{D})$	**Events of the relation $\vdash_p$ of $\mathcal{D}$**
if $p \in \mathbb{P} \setminus \mathbb{B}$, $\alpha \in \mathbb{A}$:	if $p \in P \setminus D$:
departure event: $(s, q, 0) \vdash_p^\alpha (s', q', out)$	**send event:** $\big((q,1,s), 0, \perp\big) \vdash_p \big((q',0,\sharp), out, s'\big)$
arrival event: $(s, q, in) \vdash_p^\alpha (s', q', 0)$	**receive event:** $\big((q,0,\sharp), in, s\big) \vdash_p \big((q',1,s'), 0, \perp\big)$
α **remains at** p**:** $(s, q, 0) \vdash_p^\alpha (s', q', 0)$	**internal event:** $\big((q,1,s), 0, \perp\big) \vdash_p \big((q',1,s'), 0, \perp\big)$
if $p \in \mathbb{B}$	if $p \in D$
destroy event: $(s, q_{BH}, in) \vdash_p^\alpha (s_{BH}, q_{BH}, 0)$	**ignore event:** $((q_{BH}, 0, \sharp), in, s) \vdash_p ((q_{BH}), 0, \sharp), 0, \perp)$

Table 3.3: Complexity of the Simulation of the Mobile Agent Algorithm $\mathcal{A}$ with Black Holes by Message Passing Algorithm $\mathcal{D}$ with Always Dead Processes

Message Complexity	$MSG(\mathcal{D}, C) = \sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) = TotalMove(\mathcal{A}, \mathbb{S})$
Bit Complexity	$Bit(\mathcal{D}, C) = \sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$
Time Complexity	$Time(\mathcal{D}, C) = Time\big(\mathcal{A}, \mathbb{S}\big)$
Local Space Complexity	$LocMem(\mathcal{D}, C) = \max_{p \in \mathbb{P}}\{LocMemWB(\alpha, \mathcal{A}, \mathbb{S})\} + \max_{\alpha \in \mathbb{A}}\{LocMemAg(\alpha, \mathcal{A}, \mathbb{S})\}$
Total Space Complexity	$Mem(\mathcal{D}, C) = \sum_{p \in \mathbb{P}} LocMemWB(\alpha, \mathcal{A}, \mathbb{S}) + \sum_{\alpha \in \mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$

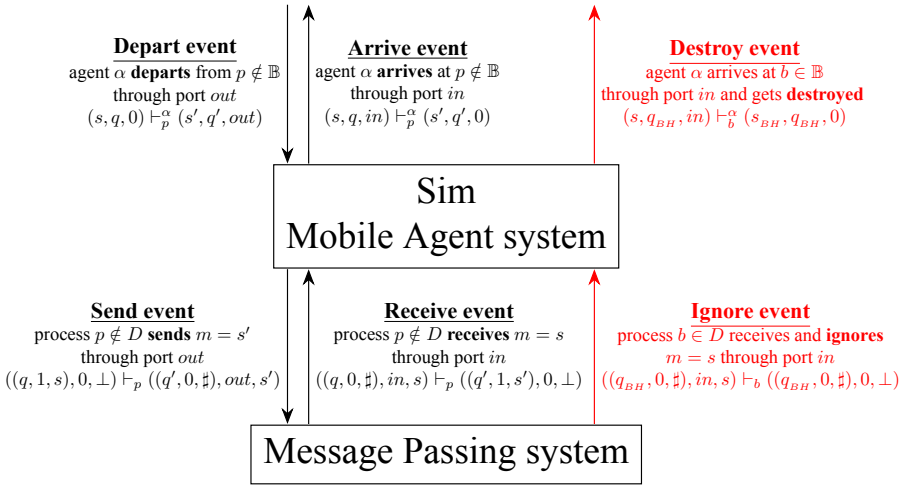

Figure 3.5: Simulation of Mobile agent system with black holes by a Message Passing system with always dead processes, where the inputs of the mobile agent system (depart events) are transformed into inputs of the message passing system (send events) and the outputs of the message passing system (receive or ignore events) are transformed into outputs of the mobile agent system (arrive or destroy events). Running the simulation algorithm on top of the message passing system with always dead processes produces the same appearance as does running the algorithm on top of the mobile agent system with black holes.

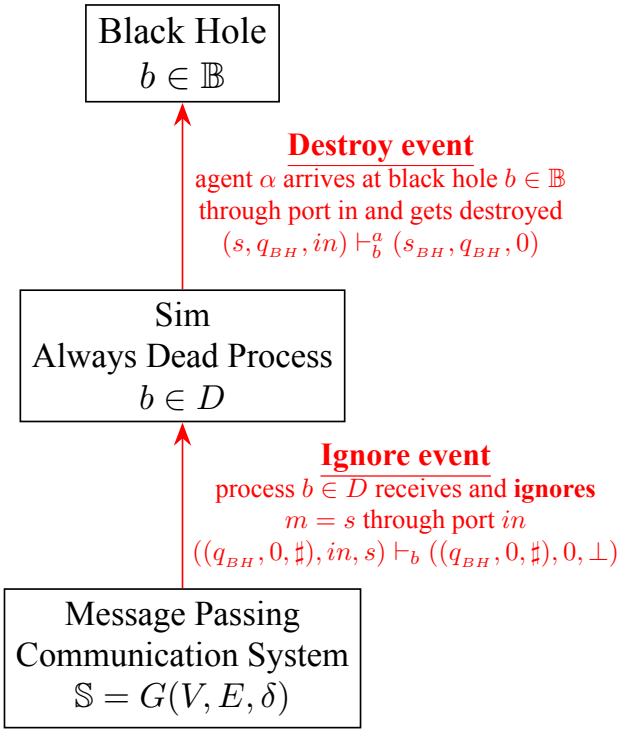

Figure 3.6: Simulation of Black Hole $b \in \mathbb{B}$ of the Mobile Agent system by always dead process $b \in D$ of the Message Passing system. The output event (ignore event) of the Message Passing communication system to the always dead process is transformed into an output event (destroy event) of the always dead process to the black hole. The simulation of the Black Hole $b \in \mathbb{B}$ by the always dead process $b \in D$ produces the same appearance to the system as the Black Hole

3.3.2 Simulation of an ADP-MP algorithm by a BH-MA algorithm

Let $\mathcal{D}$ be a message passing algorithm on the system $(P,\ C,\ D,\ \lambda)$ such that in every execution of $\mathcal{D}$ the total number of messages sent to always dead processes is at most $k-1$,for some $k \in \mathbb{N}$. If we wish to simulate algorithm $\mathcal{D}$ on top of a mobile agent system with black holes $(\mathbb{A},\ \mathbb{P},\ \mathbb{B},\ \mathbb{S},\ \pi_0, \lambda')$, where $\mathbb{S} = (V, E, \lambda')$, $|\mathbb{A}| \geq k$, and $|D| \leq |\mathbb{B}|$, the mobile agents should be able to move on the navigation subsystem and execute the local message passing algorithm $\mathcal{D}_p$ on each execution place $p \in \mathbb{P}$. Due to the presence of black holes in the system, the partial graph traversal of algorithm 2 cannot be applied, since some mobile agents might be destroyed by a black hole during the partial graph traversal of algorithm 2.

In order to overcome this problem, each agent $\alpha \in \mathbb{A}$ will execute the local message passing algorithm $\mathcal{D}_p$ on $p \in P$ and at the same time expand the partial spanning tree T_α each time they simulate the send events of algorithm $\mathcal{D}_p$ to unvisited execution places. Additionally, we have to ensure that even if an agent $\alpha \in \mathbb{A}$ is destroyed by a blackhole, the execution places $p \in T_\alpha$ are assigned to another alive agent to simulate them. For this reason we will modify algorithm 4 $AnSimulate$ of [13], to simulate a message passing algorithm $\mathcal{D}$ that sends at most $k-1$ messages to always dead processes in a mobile agent system with at least k agents and $|\mathbb{B}| \leq |D|$.

Algorithm 6 Simulation of a Message Passing algorithm $\mathcal{D}$ in a Mobile Agent System with Always Dead processes

Initially:
$traversed_u(\delta_u(v)) \leftarrow NULL$, $\forall u$, v such that $(u,v) \in E(G)$
$fatherlink_u \leftarrow NULL$, $\forall u \in V(G)$
$fatherlink_{\pi_0(\alpha)} \leftarrow -1$, $\forall \alpha \in \mathbb{A}$
$\pi(\alpha) \leftarrow \pi_0(\alpha)$, $\forall \alpha \in \mathbb{A}$
$T_\alpha \leftarrow \{\pi_0(\alpha)\}$, $\forall \alpha \in \mathbb{A}$

for Phase $i \geq 1$: **do**
Step 1: Agent $\alpha \in \mathbb{A}$ does a dfs traversal of T_α and executes the local algorithm $\mathcal{D}_u$ on each $u \in T_\alpha$
while $TBD_u \neq \emptyset$ **do**
$< u, v, m > \leftarrow TBD_u.dequeue()$
if $traversed_u(v) \neq$ "T" and $traversed_u(v) \neq$ "NT" **then**
$traversed_u(v) \leftarrow$ "$exploring$"
$\pi(\alpha) \leftarrow v$ ▷ agent α migrates to v
$in-buf_v.enqueue(< u, v, m >)$
if $fatherlink_v \neq NULL$ **then**
$traversed_v(\delta_v(u)) \leftarrow$ "NT" ▷ edge (u,v) is safe and $(u,v) \notin T_\alpha$
$\pi(\alpha) \leftarrow u$ ▷ agent α migrates to u
$traversed_u(\delta_u(v)) \leftarrow$ "NT"
else
$fatherlink_v \leftarrow \delta_v(u)$
$traversed_v(\delta_v(u)) \leftarrow$ "T" ▷ edge (u,v) is safe and $(u,v) \in T_\alpha$
$\pi_\alpha \leftarrow u$ ▷ agent α migrates to u
Agent α executes local algorithm $\mathcal{D}_v$ on v
Write $DONE(i, n_\alpha)$ on $whiteboard_p$
if $ANNEXED_{\pi_0(\alpha)} = (j, n_\beta, pathto(e'))$[1] **then**
α traverses $pathto(e')$ to arrive to node u and sets $traversed_u(e') \leftarrow T$
$n_\alpha \leftarrow n_\alpha + n_\beta$
$T_\alpha \leftarrow T_\alpha + \{e'\} + T_\beta$
$ANNEXED_{\pi_0(\alpha)} = NULL$
Go to step 3

Step 2: Agent $\alpha \in \mathbb{A}$ starts a dfs traversal of its territory T_α. During the traversal for each external edge (i.e. $e \in E$ such that $traversed_u(e) =$ "NT" for some $u \in T_\alpha$) it traverses the edge $e = \{u, v\}$, and reads $DONE(j, n_\beta)$
if $j < i$ or ($j = i$ and $n_\beta < n_\alpha$) **then**
Go to homebase of $\beta \in \mathbb{A}$ (by following fatherlinks)
if $ANNEXED_{\pi_0(\beta)} = NULL$ **then**
$ANNEXED_{\pi_0(\beta)} \leftarrow (i, n_\alpha, pathto(e)$
$n_\alpha \leftarrow n_\alpha + n_\beta$
$T_\alpha \leftarrow T_\alpha + \{e\} + T_\beta$

Step 3: Agent $\alpha \in \mathbb{A}$ updates territory T_α to include all territories it annexed and those annexed but the agents it defeated.
The fatherlink of each node in T_α is updated and n_α is modified accordingly.
if $fState_u =$ "$Finished$" $\forall u \in T_\alpha$ **then**
Execute termination detection algorithm 5
else
Go to phase $i + 1$

Proposition 3.10. Algorithm 6 of the mobile agent system $(\mathbb{P}, \mathbb{A}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$ simulates any message passing algorithm $\mathcal{D}$ defined on the system (P, C, D, λ), where $C = \mathbb{S} = (V, E, \lambda)$ and $|\mathbb{B}| \leq |D|$, given that in algorithm $\mathcal{D}$ at most $k-1$ messages are sent to always dead processes D and that the graph $G \setminus D$ is connected.

Proof. Let $\mathcal{E}_{\mathcal{A}}$ be an execution of algorithm 6 that simulates message passing algorithm $\mathcal{D}$.

Claim 1: Every $p \in P$ belongs to at least one T_α, $\alpha \in \mathbb{A}$.

proof: It suffices to show that every node will be visited by at least one mobile agent, while simulating the send events. Since the graph $G \setminus D$ is connected, we can assume without loss of generality that $\forall p \in P$ here exists a subsequence of send events of execution $\mathcal{E}$ from some homebase $\pi_0(\alpha)$, $\alpha \in \mathbb{A}$ to node p.

Claim 2: For every agent $\alpha \in \mathbb{A}$ that is destroyed, every $p \in T_\alpha$ will be assigned to an alive agent $\beta \in \mathbb{A}$.

proof: By algorithm 6, when an agent $\alpha \in \mathbb{A}$ is destroyed by a black hole or is slow then a neighboring agent will expand his territory to include T_α.

Claim 3: At least one agent $\alpha \in \mathbb{A}$ survives till the end of the execution of the simulation algorithm $\mathcal{A}$ and the union of the territories of the alive agents at the end of the algorithm is V.

proof: By algorithm $\mathcal{D}$, at most $k-1$ messages are sent to always dead processes. In the simulation an agent will traverse an unexplored edge $\{u, v\}$ only if in algorithm $\mathcal{D}_u$ u sends a message to v. Therefore at least one agent $\alpha \in \mathbb{A}$ will survive till the end of the execution of the simulation algorithm $\mathcal{A}$. Since when an agent is slow or has been destroyed, another mobile agent takes responsivility of his territory, then by the end of the algorithm the union of the territories of the alive agents contains all vertices of graph G.

$$\bigcup_{\substack{\alpha \in \mathbb{A} \\ \alpha \text{ alive} \\ \text{till the end}}} V(T_\alpha) = V(G)$$

□

Complexity

Agent Complexity

Proposition 3.11. Let $\mathcal{A}$ be the simulation algorithm 6 on the message passing algorithm $\mathcal{D}$. The minimum number of agents required for algorithm $\mathcal{A}$ to be resilient is $Agents(\alpha, \mathbb{S}) = k$ where $k-1$ is the maximum number of messages sent to always dead processes over all executions of algorithm $\mathcal{D}$.

Proof. Algorithm $\mathcal{A}$ is resilient if at least one agent survives and completes the simulation of algorithm $\mathcal{A}$.

- If $|\mathbb{A}| = k$ and $|\mathbb{B}| \leq |D|$ and the number of messages sent to always dead processes is at most $k-1$, then by the proof of proposition 3.10 at least one agent will survive and complete the simulation of algorithm $\mathcal{D}$.

[1] $pathto(e')$ is the path that contains edge labels from $\pi_0(\alpha)$ to e'

- If $|\mathcal{A}| = k-1$, $|\mathbb{B}| = |D|$ and there exists an execution of algorithm $\mathcal{D}$ that sends $k-1$ messages to always dead processes, then by running algorithm 6 all the mobile agent will be eventually destroyed by some black hole.

Therefore the minimum number of agents required is k. □

3.4 Crash Failures in MP (CF-MP) and Black$^+$ holes in MA (B$^+$H-MA)

3.4.1 MP model with Crash Failures (CF)

In the message passing model a crash failure or crash fault or stopping failure is a type of benign failure. A crashed process is a process that executed the local algorithm correctly up to a moment t_0 and after time t_0 it did not execute a single step of the algorithm. [23]. We note that an always dead process is a special case of a crash failure where the time of the crash is at the beggining of the algorithm, $t_0 = 0$.

A message passing model with t-crash failures is defined by (P, C, F, λ) where:

- P, C, λ are as defined in section 1.2
- $F \subsetneq P$, $|F| \leq t$ is a set of at most t-faulty processes.

Without loss of generality we assume that an adversary can choose to corrupt any subset $F \subsetneq P$ of at most t-processes and choose the time $t_0 : F \to \mathbb{R}$ of stopping of each of the corrupted processes.

Message Passing algorithm with Crash Failures

A message passing algorithm with t-crash failures is defined as in section 1.2 with the following modifications:

- The set Q_P of possible states of the processes contains a special terminal state c_f that indicates that a process crashed and will not execute any further event.
- The faulty processes are modelled as state machines, where the allowed events are send, receive, internal event as described in section 1.2, and additionally a stop event, which is modelled as an input action.
- At time $t_0(f)$, a stop event is applied to process $f \in F$, and f stops executing the local algorithm $\mathcal{D}_f$. The **stop event** is modelled as an input action:

  $$(c, in^*, \text{"}crash\text{"}) \vdash_f (c_f, 0, \perp)$$

  and arrives from an adversary or from an unspecified external environment through a special port in^*. In the stop event the adversary changes the current state of process $f \in F$ to c_f.
- A stop event can change only the state of process f
- When a message is sent to a crashed process we can assume without loss of generality that it is ignored, since the effect of this action is not seen outside of f,

$f \in F$. [23]. Therefore, the only applicable event for process $f \in F$ with state c_f is the **ignore event**:

$$(c_f,\, in,\, m) \vdash_f (c_f,\, 0, \perp)$$

where c_f is the state of process $f \in F$, in is the port through which the message m is received.
$M \leftarrow M \setminus \{(p', m, f)\}$ where p' is such that $\delta_f(p') = in$

- The termination property is modified to depend only to correct processes, regardless of the failures of other processes.

The interaction of a crashed process with the message passing communication system and the external environment is depicted in figure 3.7. The crashed process sends and receives messages according to the message passing algorithm until the stop event from the external environment occurs, at time t_0. After the stop event every message that arrives to the always dead process from the communication system is received and immediately ignored. Therefore without loss of generality, the interface of the message passing system contains also the ignore events of the crashed processes, as depicted in figure 3.8.

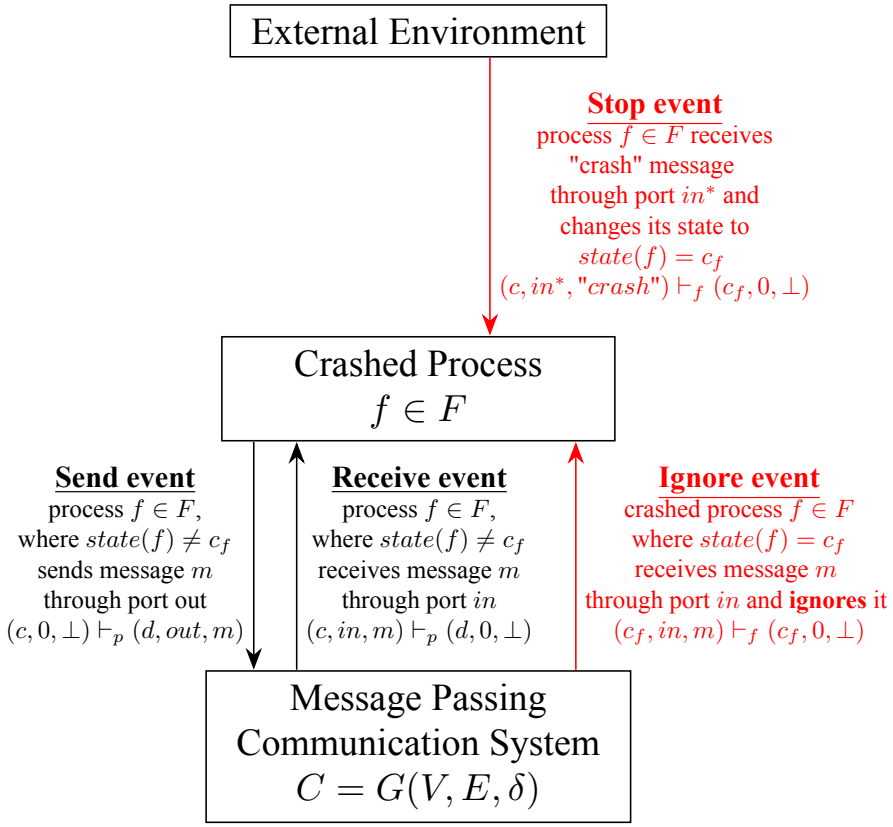

Figure 3.7: The interface of a crashed process with the message passing communication system. The crashed process $f \in F$ executes the algorithm correctly up until the **stop event** arrives from the external environment(or the adversary) through the special port in^* at time t_0, chosen by the adversary. After the stop event, any message m received by $f \in F$ from the message passing communication system is ignored and the state of the process f doesn't change.

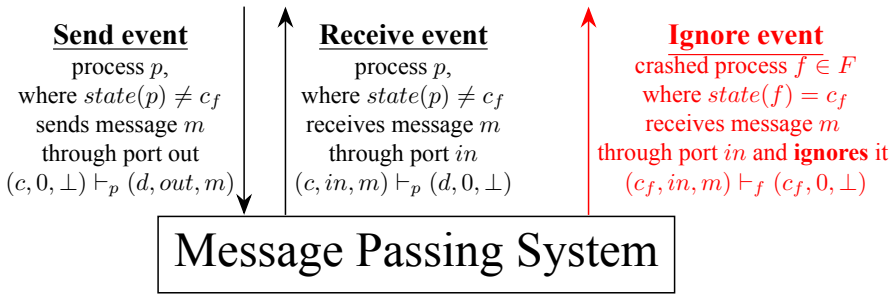

Figure 3.8: The interface of a Message passing system with crash failures. The inputs of the message passing system are the send events and the outputs are the receive events and the ignore events.

3.4.2 MA Model with Black$^+$ holes (B$^+$H)

In the mobile agent model a *black$^+$ hole* (B$^+$H) is a type of stationary host attack similar to the *black hole* model defined above. More precisely, a *black$^+$ hole* is a stationary process located at an execution place $b \in \mathbb{P}$ that destroys every agent that visits b at any time $t \geq t_0$, without leaving any trace to the surviving agents. Without loss of generality we assume that an adversary or an unspecified entity from the external environment can choose the subset $\mathbb{B}^+ \subsetneq \mathbb{P}$ of black$^+$ holes and the time $t_0 : \mathbb{B}^+ \to \mathbb{R}$ that each $b \in \mathbb{B}^+$ will start acting as a black hole. Until time $t_0(b)$, b acts as a safe execution place.

A mobile agent system with a set of black$^+$ holes is defined as $(\mathbb{A}, \mathbb{P}, \mathbb{B}^+, \mathbb{S}, \pi_0, \lambda)$ where:

- $\mathbb{A}$, $\mathbb{P}$, $\mathbb{S}, \lambda$ are defined as in section 1.3.
- $\mathbb{B}^+ \subsetneq \mathbb{P}$ is a set of black$^+$ holes that destroy any visiting agent without leaving any trace.
- $\mathbb{P} \setminus \mathbb{B}^+$ is the set of safe execution places
- $\pi_0 : \mathbb{A} \to \mathbb{P} \setminus \mathbb{B}^+$ agents are initially located on safe execution places.

Black hole model is a special case of the Black$^+$ hole model when $t_0 = 0$

Mobile Agent Algorithm with a black$^+$ hole

A mobile agent algorithm with a set of black$^+$ holes is defined as in section 1.3 with the following modifications:

- The set $Q_{\mathbb{A}}$ of possible states of the agents contains a special terminal state $s_{_{BH^+}}$ that indicates that an agent died when visiting a black hole $b \in \mathbb{B}^+$.
- The set $Q_{\mathbb{P}}$ of possible states of the execution places contains a special state $q_{_{BH^+}}$ that indicates that the execution place destroys any incoming agent.
- At time $t_0(b)$, $b \in \mathbb{B}^+$ starts acting as a black hole. Without loss of generality we assume that the adversary changes the state of the execution place to $q_{_{BH^+}}$ by invoking the **black$^+$ hole event**. The **black$^+$ hole event** modelled as an input action from the external environment as follows:

  $$("black^+", q, in^*) \vdash_b (\sharp, q_{_{BH^+}}, 0)$$

  where "$black^+$" is a special state indicating the change of state of the execution place, $q \in Q_{\mathbb{P}} \setminus \{q_{_{BH^+}}\}$ is the old state of $b \in \mathbb{B}^+$, $\sharp$ is a null state, $q_{_{BH^+}}$ is the new state of $b \in \mathbb{B}^+$ and in^* is a special port through which the black$^+$ hole event arrives from the external environment.
- When agent $\alpha \in \mathbb{A}$ visits a black$^+$ hole $b \in \mathbb{B}^+$ at time $t \geq t_0(b)$ the only applicable event is the **destroy event**:

  $$(s, q_{_{BH^+}}, in) \vdash_b^{\alpha} (s_{_{BH^+}}, q_{_{BH^+}}, 0)$$

  where s is the old state of agent $\alpha \in \mathbb{A}$, $q_{_{BH^+}}$ is old state of $b \in \mathbb{B}^+$, in is the port through which α arrived to b, $s_{_{BH^+}}$ is the new and terminal state of α and there is no other applicable event that involves agent $\alpha \in \mathbb{A}$

- after a destroy event of agent $\alpha \in \mathbb{A}$ on black hole $b \in \mathbb{B}^+$:
  $\mathbb{M} \leftarrow \mathbb{M} \setminus \{(p', \alpha, p)\}$ where $p' \in \mathbb{P}$ is such that $\delta_p(p') = in$
  $\pi(\alpha) \leftarrow \perp$
  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{\alpha\}$

- The termination property is modified to depend only to the surviving agents, i.e. agents that are not destroyed by the black$^+$ hole.

The interaction of a Black$^+$ Hole with the mobile agent navigation system and the external environment is depicted in figure 3.9. The Black$^+$ Hole behaves as a non malicious execution place up until the time that the **Black$^+$ Hole event** occurs. Every agent $\alpha \in \mathbb{A}$ that arrives to the from the navigation system is destroyed without leaving any trace. Therefore without loss of generality, the interface of the mobile agent system contains also the destroy events of the Black$^+$ Hole, as depicted in figure 3.10.

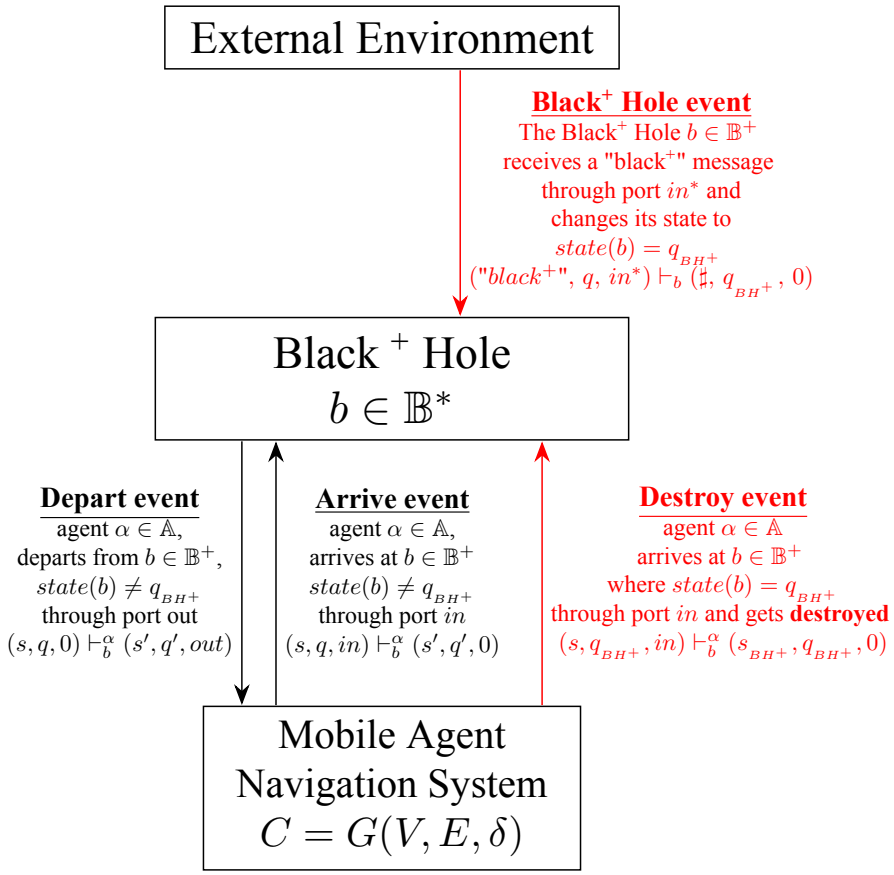

Figure 3.9: The interface of a Black$^+$ Hole with the mobile agent navigation system. The inputs of the Black$^+$ Hole are the **arrive events**, and the outputs are the **send events**. After the **Black$^+$ Hole event** occurs as input from the external environment the inputs of the Black$^+$ Hole are only the destroy events.

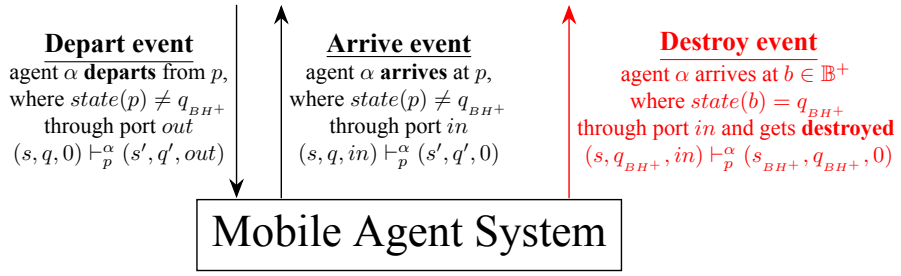

Figure 3.10: The interface of a Mobile agent system with black⁺ holes. The inputs of the mobile agent system are the depart events and the outputs are the arrive events and the destroy events.

3.5 Simulation of a B⁺H-MA algorithm by a CF-MP algorithm

A mobile agent algorithm with black⁺ holes can be simulated in the message passing system with crash failures. The simulation is very similar with the simulation presented in 3.3.1 if we replace the always dead processes with crash failure and assume that the mobile agent algorithm $\mathcal{A}$ tolerates black⁺holes. If in the mobile agent algorithm $\mathcal{A}$ agent $\alpha \in \mathbb{A}$ visits the black⁺hole $b \in \mathbb{B}^+$ after time $t_0(b)$ it is destroyed without leaving any trace, while in the simulation in the message passing system when a message is sent to a crashed process $b \in F$ after time $t_0(b)$ it is destroyed and the sender has no evidence that the receiver b is faulty. Therefore agents don't know if agent α has died or delayed to return from visiting node u, while in the simulation, neighbouring processes don't know if process u is faulty or has delayed to respond.

Let $(\mathbb{A},\ \mathbb{P},\ \mathbb{B}^+,\ \mathbb{S},\ \pi_0,\ \lambda)$ be a mobile agent system with a set of black⁺ holes $\mathbb{B}^+$, where $\mathbb{S} = (V, E, \delta)$ is the navigation subsystem and $\mathcal{A} = (\vdash_p^\alpha)_{\alpha \in \mathbb{A},\ p \in \mathbb{P}}$. The mobile agent algorithm $\mathcal{A}$ can be simulated in the message passing system $(P,\ C,\ F,\ \lambda')$ in algorithm $\mathcal{D}$ as in section 2.1 with the following modifications:

- Each non faulty process $p \in P \setminus F$ of the message passing system corresponds to a safe execution place $p \in \mathbb{P} \setminus \mathbb{B}^+$ of the mobile agent system and executes algorithm $\mathcal{D}_p$ as defined in section 2.1.
- Each crash failure process $b \in F$ of the message passing system corresponds to a black⁺ hole $b \in \mathbb{B}^+$.
- The **ignore event** of a token $t(\alpha)$ by a crashed process:

  $$(c_f,\ in,\ s) \vdash_b (c_f,\ 0,\ \perp)$$

  where $c_f = (q_{BH^+},\ 0,\ \sharp)$, and in is the port through which token $t(\alpha)$ with state s is received, has similar effect with the **destroy event** of an agent α by black⁺ hole $b \in \mathbb{B}^+$:

  $$(s, q_{BH^+}, in) \vdash_b^\alpha (s_{BH^+}, q_{BH^+}, 0)$$

  since after the ignore event token $t(\alpha)$ will not be involved in any other event and so will agent agent α after the destroy event.

Proposition 3.12. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm implemented on the mobile agent system with black$^+$ holes $(\mathbb{A},\, \mathbb{P},\, \mathbb{B}^+,\, \mathbb{S},\, \pi_0,\, \lambda)$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system with crash failures $(P,\, C,\, F,\, \lambda')$. Then algorithm $\mathcal{D}$ of the message passing system simulates algorithm $\mathcal{A}$ of the mobile agent system.

Proof. Similarly with the proof of proposition 3.5 with the additional remarks that:

1. $|F| \leq |\mathbb{B}^+|$

2. there exists an execution $\mathcal{E}_\mathcal{A}$ of $\mathcal{A}$ that the time that a process $p \in F$ crashes is the same as the time that the corresponding execution place $p \in \mathbb{B}^+$ becomes a black hole.

□

Simulation of B$^+$H by a crashed process

Black $_+$ holes can be simulated by crash failures. In this case, the **black$^+$ hole event**, that an execution place becomes a black$^+$ hole:

$$(\text{"}black\text{"},\, q,\, in^*) \vdash_b (\sharp,\, q_{_{BH^+}},\, 0)$$

is simulated by a **stop event**, that a process b crashes:

$$((q,\, 0,\, \sharp),\, in^*, \text{"}crash\text{"}) \vdash_b ((q_{_{BH^+}},\, 0,\, \sharp),\, 0,\, \perp)$$

Therefore, the **destroy event** of an agent α by a black$^+$ hole b:

$$(s,\, q_{_{BH^+}},\, in) \vdash_b^\alpha (s_{_{BH^+}},\, q_{_{BH^+}},\, 0)$$

is simulated as an **ignore event** of token $t(\alpha)$

$$((q_{_{BH^+}},\, 0,\, \sharp),\, in,\, s) \vdash_b ((q_{_{BH^+}},\, 0,\, \sharp),\, 0, \perp)$$

Again, after the destroy event agent α will not be involved in any other event and token $t(\alpha)$ will be ignored and no other event will involve token $t(\alpha)$.

Termination

Lemma 3.13. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm on the mobile agent system with black$^+$ holes implemented $(\mathbb{A},\, \mathbb{P},\, \mathbb{B}^+,\, \mathbb{S},\, \pi_0,\, \lambda)$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system with crash failures $(P,\, C,\, F,\, \lambda')$. If algorithm $\mathcal{A}$ has the termination property then algorithm $\mathcal{D}$ has the termination property.

Proof. Similar with the proof of lemma 3.6

□

Complexity

Message Complexity

Proposition 3.14. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm that tolerates t black$^+$ holes implemented on the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{B}^+,\, \mathbb{S},\, \pi_0,\, \lambda)$, where $|\mathbb{B}^+| \leq t$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system with crash failures $(P,\, C,\, F,\, \lambda')$, where $|F| \leq t$ and $C = \mathbb{S}$. Then the **total number of messages exchanged** during the execution of $\mathcal{D}$ is:

$$MSG(\mathcal{D}, C) = \sum_{\alpha\in\mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) = TotalMove(\mathcal{A}, \mathbb{S})$$

where $Move(\alpha, \mathcal{A}, \mathbb{S})$ is the total number of moves of agent $\alpha \in \mathbb{A}$ in the execution of $\mathcal{A}$. The total number of messages sent to the set of faulty processes F is at most $k-1$, where $k = |\mathbb{A}|$.

The **total size of messages exchanged** is

$$Bit(\mathcal{D}, C) = \sum_{\alpha\in\mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$$

where $LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$ is the local space complexity of agent $\alpha \in \mathbb{A}$, i.e. the size of notebook of α.

Proof. If in the simulation algorithm $\mathcal{D}$ there is a send event for process $p \in P \setminus F$ and token $t(\alpha)$, $\alpha \in \mathbb{A}$ then in algorithm $\mathcal{A}$ agent $\alpha \in \mathbb{A}$ departs from the execution place $p \in \mathbb{P} \setminus \mathbb{B}^+$. The size of each message $t(\alpha)$ in $\mathcal{D}$ equals the size of the $notebook_\alpha$ of the corresponding agent α or $state(\alpha)$.

If in the simulation algorithm $\mathcal{D}$ there is a send event to a crashed process $b \in D$, then in algorithm $\mathcal{A}$ agent $\alpha \in \mathbb{A}$ arrives to black$^+$ hole $b \in \mathbb{B}^+$ and is destroyed. □

Time Complexity

Proposition 3.15. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm that tolerates t black$^+$ holes implemented on the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{B}^+,\, \mathbb{S},\, \pi_0,\, \lambda)$, where $|\mathbb{B}^+| \leq t$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system with crash failures $(P,\, C,\, F,\, \lambda')$, where $|F| \leq t$ and $C = \mathbb{S}$. Then the **time complexity** of algorithm $\mathcal{D}$ is:

$$Time(\mathcal{D}, C) = Time(\mathcal{A}, \mathbb{S})$$

where $Time(\mathcal{A}, \mathbb{S})$ is the time complexity of mobile agent algorithm $\mathcal{A}$ on the navigation system $\mathbb{S}$.

Proof. The proof is similar with the proof of proposition 2.4 □

Space Complexity

Proposition 3.16. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm that tolerates t black$^+$ holes implemented on the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{B}^+,\, \mathbb{S},\, \pi_0,\, \lambda)$, where $|\mathbb{B}^+| \leq t$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system with crash failures $(P,\, C,\, F,\, \lambda')$, where $|F| \leq t$ and $C = \mathbb{S}$ and $C = \mathbb{S}$. Then the **local space complexity** of algorithm $\mathcal{D}$ is:

$$LocMem(\mathcal{D}, C) = \max_{p\in\mathbb{P}} LocMemWB(p, \mathcal{A}, \mathbb{S}) + \max_{\alpha\in\mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$$

where $LocMemWB(p, \mathcal{A}, \mathbb{S})$ is the local space complexity of execution place $p \in \mathbb{P}$ and $locMemAg(\alpha, \mathcal{A}, \mathbb{S})$ is the local space complexity of agent $\alpha \in \mathbb{A}$ of algorithm $\mathcal{A}$ on the navigation system $\mathbb{S}$. The **total space complexity** of algorithm $\mathcal{D}$ is:

$$Mem(\mathcal{D}, C) = \sum_{p \in \mathbb{P}} LocMemWB(p, \mathcal{A}, \mathbb{S}) + \sum_{\alpha \in \mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$$

Proof. The proof is similar with the proof of proposition 2.5 □

The simulation presented is summarised in tables 3.4, 3.5 an figure 3.11. The simulation of a Black$^+$ Hole by a Crashed process is depicted in figure 3.12. The complexity of the simulation is summarised in table 3.6.

Table 3.4: Simulation of Mobile Agent Algorithm $\mathcal{A}$ with black$^+$ holes by Message Passing Algorithm $\mathcal{D}$ with crash failures

Mobile Agent Model with black$^+$ holes	**Message Passing Model with crash failures**
System: $(\mathbb{A},\ \mathbb{P},\ \mathbb{B}^+,\ \mathbb{S},\ \pi_0,\ \lambda)$	**System:** $(P,\ C,\ F,\ \lambda')$
$\mathbb{S} = (V,\ E,\ \delta)$ navigation subsystem $\mathbb{P}$ or V : execution places, equipped with a whiteboard E : migration ports $\delta_u : N_G(u) \to [1,\ deg_G(u)] :\ u \in V$ port labelling function λ : initial states of places and agents	$C = (V,\ E,\ \delta)$ communication subsystem P or V : asynchronous processes E : communication channels $\delta_u : N_G(u) \to [1,\ deg(u)] :\ u \in V$ port labelling function λ' : initial states of processes $\lambda'(p) = \begin{cases} (\lambda(p),\ 1,\ \lambda(\alpha)), & \text{if } \pi_0(p) = \alpha \\ (\lambda(p),\ 0,\ \sharp), & \text{otherwise} \end{cases}, p \in P$
$\mathbb{A}$: set of k asynchronous mobile agents. $\pi_0 :\ \mathbb{A} \to V$ initial placement of agents. $\pi :\ \mathbb{A} \to V$ location of agents.	k tokens π_0 : initial placement of tokens. π : location of tokens.
Black$^+$ hole $b \in \mathbb{B}^+ \subsetneq \mathbb{P}$**:** a stationary process that destroys any incoming agent after time $t_0(b)$ $s_{_{BH^+}}$: the state of the destroyed agent	**Faulty process** $b \in F \subsetneq P$**:** a process that does not execute a single step of the algorithm after time $t_0(b)$ $state(b) = s_c = (q_{_{BH^+}}, 0, \sharp)$ after time $t_0(b)$
Mobile agent algorithm $\mathcal{A}$ $Q_{\mathbb{A}}$: set of possible states of $\alpha \in \mathbb{A}$ $Q_{\mathbb{P}}$: set of possible states of $p \in \mathbb{P}$ $I_{\mathbb{A}} \subseteq Q_{\mathbb{A}} \setminus \{s_{_{BH^+}}\}$: set of initial states of α $I_{\mathbb{P}} \subseteq Q_{\mathbb{P}}$: set of initial states of $p \in \mathbb{P}$ $state^{\mathcal{A}}(p),\ p \in \mathbb{P}$ $state^{\mathcal{A}}(\alpha),\ \alpha \in \mathbb{A}$ $\mathbb{M}$: multiset of agents in transit	**Simulation message passing algorithm** $\mathcal{A}$ $\mathcal{M} = Q_{\mathbb{A}} \setminus \{s_{_{BH^+}}\}$: set of possible messages $Q = Q_{\mathbb{P}} \times \{0,1\} \times Q_{\mathbb{A}} \cup \{\sharp\} \setminus \{s_{_{BH^+}}\}$: set of possible states of $p \in P$ $I = I_{\mathbb{P}} \times \{0,1\} \times I_{\mathbb{A}} \cup \{\sharp\}$: set of initial states of $p \in P$ $state(p) =$ $= \begin{cases} (state^{\mathcal{A}}(p),\ 1,\ state^{\mathcal{A}}(\alpha)), & \pi(p) = \alpha,\ p \in F \\ (state^{\mathcal{A}}(p),\ 0,\ \sharp), & \text{otherwise} \end{cases}$ $M = \mathbb{M}$: multiset of messages in transit

Table 3.5: Simulation of Mobile Agent Algorithm $\mathcal{A}$ with black$^+$ holes by a Message Passing Algorithm $\mathcal{D}$ with crash failures

Mobile Agent Model with black$^+$ holes	**Message Passing Model with crash faults**
Events of the relation $\vdash_p^\alpha$ of $\mathcal{A}$ $h: events(\mathcal{A}) \to events(\mathcal{D})$	**Events of the relation $\vdash_p$ of $\mathcal{D}$**
if $p \in \mathbb{P} \setminus \mathbb{B}^+$, $\alpha \in \mathbb{A}$:	if $p \in P \setminus F$:
departure event: $(s,\, q,\, 0) \vdash_p^\alpha (s',\, q',\, out)$	**send event:** $\big((q,1,s),\, 0,\, \perp\big) \vdash_p \big((q',0,\sharp),\, out,\, s'\big)$
arrival event: $(s,\, q,\, in) \vdash_p^\alpha (s',\, q',\, 0)$	**receive event:** $\big((q,0,\sharp),\, in,\, s\big) \vdash_p \big((q',1,s'),\, 0,\, \perp\big)$
α **remains at** p**:** $(s,\, q,\, 0) \vdash_p^\alpha (s',\, q',\, 0)$	**internal event:** $\big((q,1,s),\, 0,\, \perp\big) \vdash_p \big((q',1,s'),\, 0,\, \perp\big)$
if $p \in \mathbb{B}^+$ and time $t \geq t_0(p)$ $state^{\mathcal{A}}(p) = q_{BH^+}$	if $p \in F$ and time $t \geq t_0(p)$ $state(p) = (q_{BH^+},\, 0,\, \sharp)$
and the only applicable event is the **destroy event:** $(s,\, q_{BH_+},\, in) \vdash_p^\alpha (s_{BH^+},\, q_{BH_+}, 0)$	and the only applicable event is the **ignore event:** $\big((q_{BH^+},0,\sharp),\, in,\, s\big) \vdash_p \big((q_{BH^+},0,\sharp),\, 0,\, \perp\big)$

Table 3.6: Complexity of the Simulation of the Mobile Agent Algorithm with Black$^+$ Holes $\mathcal{A}$ by Message Passing Algorithm $\mathcal{D}$ with crashed processes

Message Complexity	$MSG(\mathcal{D}, C) = \sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) = TotalMove(\mathcal{A}, \mathbb{S})$
Bit Complexity	$Bit(\mathcal{D}, C) = \sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$
Time Complexity	$Time(\mathcal{D}, C) = Time\big(\mathcal{A}, \mathbb{S}\big)$
Local Space Complexity	$LocMem(\mathcal{D}, C) = \max_{p \in \mathbb{P}}\{LocMemWB(\alpha, \mathcal{A}, \mathbb{S})\} + \max_{\alpha \in \mathbb{A}}\{LocMemAg(\alpha, \mathcal{A}, \mathbb{S})\}$
Total Space Complexity	$Mem(\mathcal{D}, C) = \sum_{p \in \mathbb{P}} LocMemWB(\alpha, \mathcal{A}, \mathbb{S}) + \sum_{\alpha \in \mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$

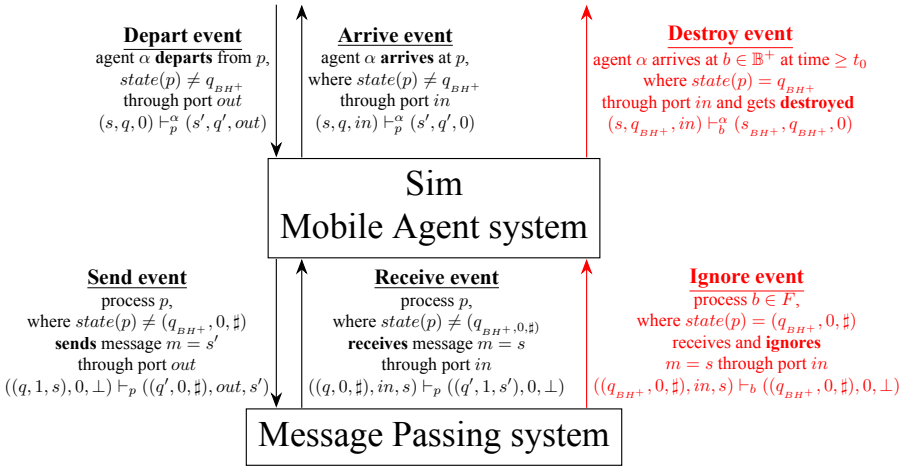

Figure 3.11: Simulation of Mobile agent system with black$^+$ holes by a Message Passing system with crash failures, where the inputs of mobile agent system (depart events) are transformed into inputs of the message passing system (send events) and the outputs of the message passing system (receive or ignore events) are transformed into outputs of the mobile agent system (arrive or destroy events), while the ignore events are transformed into destroy events. Running the simulation algorithm on top of the message passing system with crash failures produces the same appearance as does running the algorithm on top of the mobile agent system with black$^+$ holes.

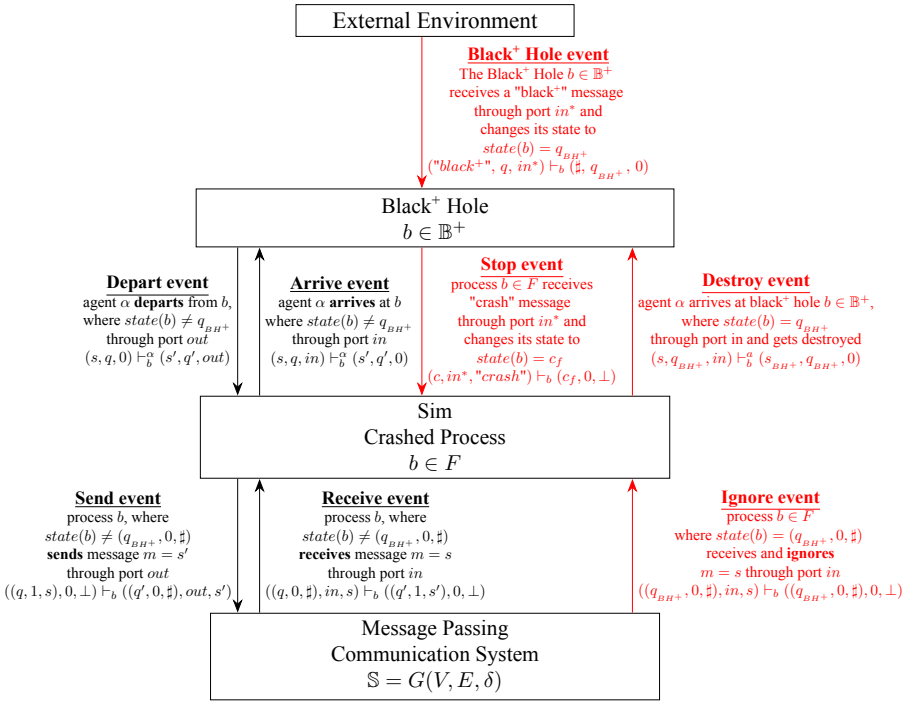

Figure 3.12: Simulation of black$^+$ holes by crashed processes.
The input event of the black$^+$ hole (black$^+$ hole event) is transformed to input event of the crashed process (crash event) and the output event of the crashed process (ignore event) is transformed to output event of the black$^+$ hole (destroy event) . The simulation of black$^+$ hole by crashed process produces the same appearance to the system as the black$^+$ hole

The simulation of a mobile agent algorithm $\mathcal{A}$ to a message passing algorithm $\mathcal{D}$, for a mobile agent system to a message passing system as described above is be the similar to the simulation in section 2.1. Most problems in the mobile agent system with a black$^+$ hole have the requirement that at least one agent survives. Therefore, the meaning of the simulation in the message passing model is to achieve a specific goal, but send up to $k - 1$ messages to the faulty process.

3.6 Omission Failures in MP (OF-MP) and Gray holes in MA (GH-MA)

3.6.1 MP model with Omission Failures (OF)

In the message passing model an omission failure (OF) is a type of benign failure, that was introduced in [28]. A process with omission failure might skip some steps of the algorithm and then continue the execution of the algorithm. [32]. We note that an always dead processes and crash failures are special cases of omission failures, where the process skips all the steps of the algorithm after time t_0.

A message passing model with t-omission failures is defined by $(P,\ C,\ O,\ \lambda)$ where:

- P, C, λ are as defined in section 1.2
- $O \subsetneq P$, $|O| \leq t$ is a set of at most t-faulty processes.

Without loss of generality we assume that an adversary can choose to corrupt any subset $O \subsetneq P$ of at most t-processes and choose which steps of the algorithm the faulty process will omit.

Message Passing algorithm with Omission Failures

A message passing algorithm with t-omission failures is defined as in section 1.2 with the following modifications:

- The set Q_P of possible states of the processes contains a special state c_o that indicates that a process crashed temporarily and is not currently executing the steps of the algorithm.
- The faulty processes are modelled as state machines, where the allowed events are send, receive, internal event as described in section 1.2, and additionally a stop and a restart event, which are modelled as input actions.
- The **stop event** is modelled as an input action:

  $$(c, in^*, \text{"}crash\text{"}) \vdash_f (c_o, 0, \perp)$$

  and arrives from an adversary or from an unspecified external environment through a special port in^*. In the stop event the adversary changes the current state of process $f \in O$ to c_o. A stop event can change only the state of process $f \in O$.
- When a message is sent to a crashed process we can assume without loss of generality that it is ignored, since the effect of this action is not seen outside of f, $f \in F$. [23]. Therefore, the only applicable event for process $f \in F$ with state c_f is the **ignore event**:

  $$(c_o, in, m) \vdash_f (c_o, 0, \perp)$$

  where c_o is the state of process $f \in F$, in is the port through which the message m is received.
  $M \leftarrow M \setminus \{(p', m, f)\}$ where p' is such that $\delta_f(p') = in$.
- The **resume event** is modelled as an input action:

  $$(c_o, in^*, \text{"}resume\text{"}) \vdash_f (c, 0, \perp)$$

  and arrives from an adversary or from an unspecified external environment through a special port in^*. In the stop event the adversary changes the current state c_o of process $f \in O$ to $c \in Q_P \setminus \{c_o\}$. A resume event can change only the state of process $f \in O$.
- The termination property is modified to depend only to correct processes, regardless of the failures of other processes.

The interaction of an omission failure with the message passing communication system and the external environment is depicted in figure 3.13. The crashed process sends and receives messages according to the message passing algorithm until the stop event from the external environment occurs. After the stop event every message that arrives to the always dead process from the communication system is received and immediately ignored. When the resume event arrives from the external environment, the state of the process changes to the state of the process before the stop event and the process continues executing the message passing algorithm correctly. The adversary can send the "stop" and "resume" events to the process repeatedly, according to any strategy. Therefore without loss of generality, the interface of the message passing system contains also the ignore events of the omission failures, as depicted in figure 3.14.

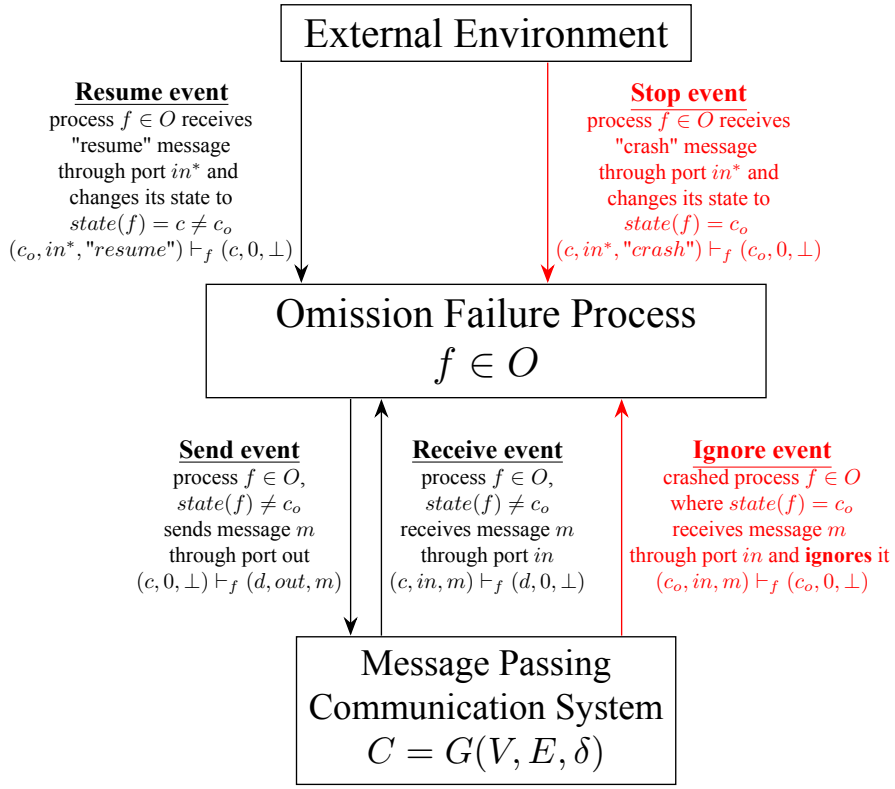

Figure 3.13: The interface of an omission failure process with the message passing communication system. The faulty process $f \in O$ executes the algorithm correctly up until the **stop event** arrives from the external environment(or the adversary) through the special port in^*. After the stop event, any message m received by $f \in F$ from the message passing communication system is ignored and the state of the process f doesn't change. When a **resume event** arrives from the external environment through the special port in^* the process continues to execute the algorithm and send and receive messages from and to the communication system.

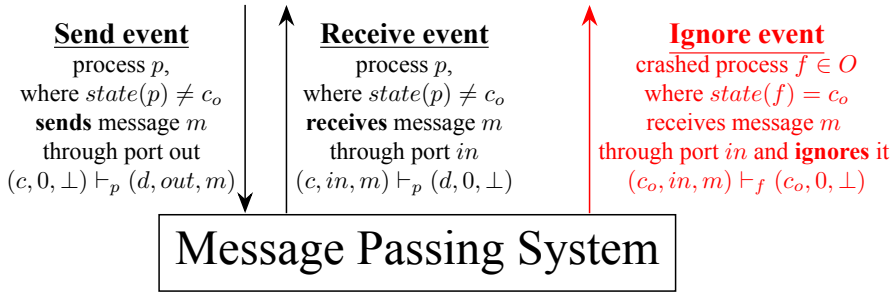

Figure 3.14: The interface of a Message Passing system with omission failures. The inputs of the message passing system are the send events and the outputs are the receive events and the ignore events.

3.6.2 MA Model with Gray holes (GH)

In the mobile agent model a *gray hole* (GH) is a type of stationary host attack similar to the *black hole* model. A *gray hole* is a stationary process located at an execution place $g \in \mathbb{P}$ that can choose whether and when to act as a black hole or a safe node. Without loss of generality we assume that an adversary or an unspecified entity from the external environment can choose the subset $\mathbb{G} \subsetneq \mathbb{P}$ of gray holes and when each $g \in \mathbb{G}$ will act as a black hole or a safe node.

A mobile agent system with a set of gray holes is defined as $(\mathbb{A}, \mathbb{P}, \mathbb{G}, \mathbb{S}, \pi_0, \lambda)$ where:

- $\mathbb{A}$, $\mathbb{P}$, $\mathbb{S}$, λ are defined as in section 1.3.
- $\mathbb{G} \subsetneq \mathbb{P}$ is a set of gray holes that can choose whether to behave as black holes or safe execution places.
- $\mathbb{P} \setminus \mathbb{G}$ is the set of safe execution places.
- $\pi_0 : \mathbb{A} \to \mathbb{P} \setminus \mathbb{G}$ agents are initially located on safe execution places.

Black$^+$ hole model, and subsequently the black hole model, is a special case of the Gray hole model, if the malicious host behaves only as a black hole at any time $t \geq t_0$

Mobile Agent Algorithm with a gray hole

A mobile agent algorithm with a set of gray holes is defined as in section 1.3 with the following modifications/additions:

- The set $Q_{\mathbb{A}}$ of possible states of the agents contains a special terminal state s_{GH} that indicates that an agent was destroyed when visiting a gray hole $g \in \mathbb{G}$
- The set $Q_{\mathbb{P}}$ of possible states of the execution places contains a special state q_{GH} that indicates that the execution place destroys any incoming agent.
- Without loss of generality, we assume that the decision of a gray hole to act as a black hole or a safe node is modelled as an input action from the external environment. The **gray hole event** is modelled as:

$$(\text{"}gray\text{"}, q, in^*) \vdash_g (\sharp, q_{GH}, 0)$$

where "$gray$" is a state that indicates that the execution place will start destroying any incoming agent, $\sharp$ is a null state, $q \in Q_{\mathbb{P}} \setminus \{q_{_{GH}}\}$ is the old state of $g \in \mathbb{G}$, $q_{_{GH}}$ is the new state of $g \in \mathbb{G}$ and in^* is a special port through which the gray hole event arrives from the external environment.
The **safe node event**, that a gray hole $g \in \mathbb{G}$ with state $q_{_{GH}}$ becomes a safe execution place is modelled as:

$$(\text{"}safe\text{"},\, q_{_{GH}},\, 0) \vdash_g (\sharp,\, q',0)$$

where "$safe$" is a state that indicates that the gray hole will act as a safe node, $\sharp$ is a null state, $q_{_{GH}}$ is old state of $g \in \mathbb{G}$ and $q' \in Q_{\mathbb{P}} \setminus \{q_{_{GH}}\}$.

- When agent $\alpha \in \mathbb{A}$ visits a gray hole $g \in \mathbb{G}$ with state $q_{_{GH}}$ the only applicable event is the **destroy event**:

$$(s,\, q_{_{GH}},\, in) \vdash_g^{\alpha} (s_{_{GH}},\, q_{_{GH}}, 0)$$

where s is the old state of agent $\alpha \in \mathbb{A}$, $q_{_{GH}}$ is old state of $g \in \mathbb{G}$, in is the port through which α arrived to g, $s_{_{GH}}$ is the new and terminal state of α and there is no other applicable event that involves agent $\alpha \in \mathbb{A}$

- after a destroy event of agent $\alpha \in \mathbb{A}$ on gray hole $g \in \mathbb{G}$:
$\mathbb{M} \leftarrow \mathbb{M} \setminus \{(p', \alpha, p)\}$ where $p' \in \mathbb{P}$ is such that $\delta_p(p') = in$
$\pi(\alpha) \leftarrow \perp$
$\mathbb{A} \leftarrow \mathbb{A} \setminus \{\alpha\}$

- The termination property is modified to depend only to the surviving agents, i.e. agents that are not destroyed by the gray hole.

The interaction of a Gray Hole with the mobile agent navigation system and the external environment is depicted in figure . The Gray Hole behaves as a non malicious execution place up until the time that the Gray Hole event occurs. After the "Gray Hole event", every agent$\alpha \in \mathbb{A}$ that arrives to the Gray Hole $g \in \mathbb{G}$ from the navigation system is destroyed without leaving any trace. When the "Safe Node event" arrives from the external environment, the gray hole $g \in \mathbb{G}$ behaves again as a safe execution place. The adversary can invoke the "Gray Hole event" and the "safe node event" according to any strategy. Therefore without loss of generality, the interface of the mobile agent system contains also the destroy events of the Black+ Hole, as depicted in figure 3.16

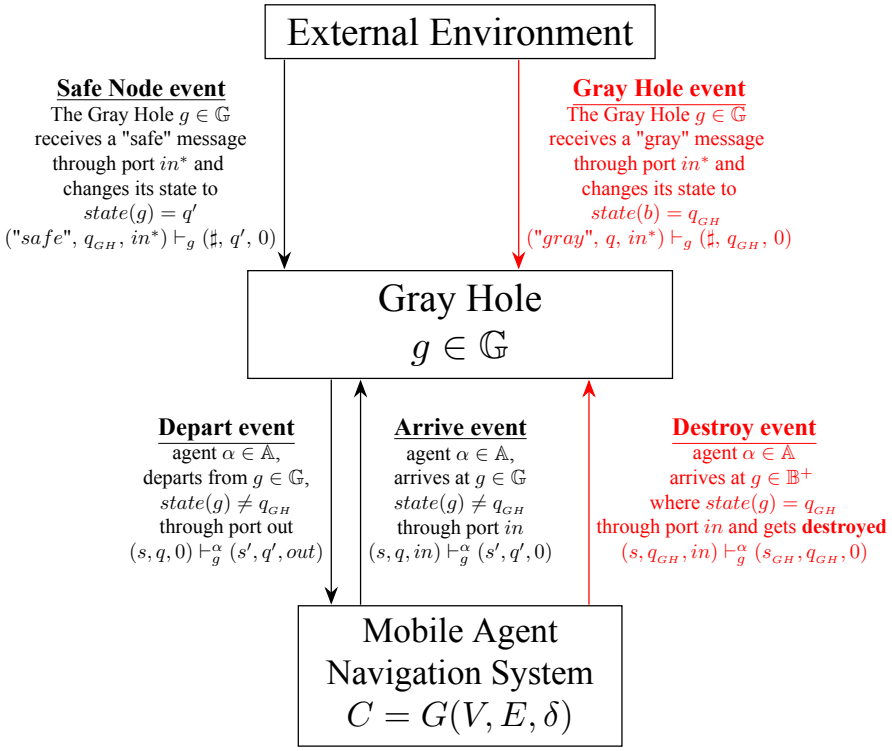

Figure 3.15: The interface of a Gray Hole with the mobile agent navigation system. The inputs of the Gray Hole from the navigation system are the arrive events and the destroy events, and the outputs are the send events. The inputs of the Gray Hole from the external environment are the "safe node event" and the "gray hole event". After the "Gray Hole event" occurs as input from the external environment the inputs of the Gray Hole from the navigation system are only the destroy events, and after the "safe node event" the inputs are the arrive events and the outputs are the depart

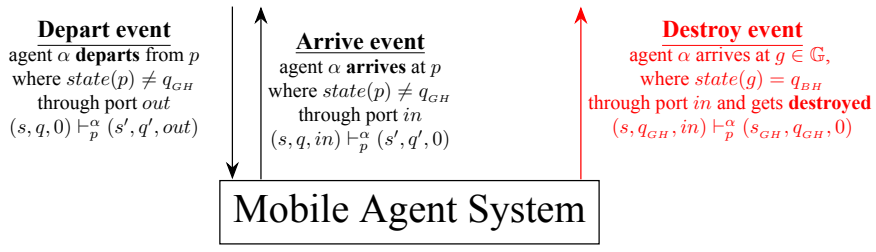

Figure 3.16: The interface of a Mobile agent system with gray holes. The inputs of the mobile agent system are the depart events and the outputs are the arrive events and the destroy events.

3.7 Simulation of a GH-MA algorithm by an OF-MP algorithm

Similarly with sections 3.3.1 and 3.5 a gray hole resilient mobile agent algorithm can be simulated in the message passing model with omission failures. In the simulation if in the mobile agent algorithm $\mathcal{A}$ agent $\alpha \in \mathbb{A}$ visits the gray hole hole $g \in \mathbb{G}$ that acts currently as black hole, it is destroyed without leaving any trace, while in the simulation in the message passing system when a message is sent to a process with omission failure $g \in O$, the receiving event will be omitted and the sender has no evidence that the receiver g is faulty. Therefore agents don't know if agent α has died or delayed to return from visiting node g, while in the simulation, neighbouring processes don't know if process g received the message or has delayed to respond.

Let $(\mathbb{A},\, \mathbb{P},\, \mathbb{G},\, \mathbb{S},\, \pi_0,\, \lambda)$ be a mobile agent system with a set of gray holes $\mathbb{G}$, where $\mathbb{S} = (V, E, \delta)$ is the navigation subsystem and $\mathcal{A} = (\vdash_p^{\alpha})_{\alpha \in \mathbb{A},\, p \in \mathbb{P}}$. The mobile agent algorithm $\mathcal{A}$ can be simulated in the message passing system $(P,\, C,\, O,\, \lambda')$ in algorithm $\mathcal{D}$ as in section 2.1 with the following modifications:

- Each non faulty process $p \in P \setminus O$ of the message passing system corresponds to a safe execution place $p \in \mathbb{P} \setminus \mathbb{G}$ of the mobile agent system and executes algorithm $\mathcal{D}_p$ as defined in section 2.1.
- Each omission failure process $g \in O$ of the message passing system corresponds to a gray hole $g \in \mathbb{G}$.
- The **ignore event** of $token(\alpha)$ by a faulty process $g \in O$:

  $$(c_f,\, in,\, s) \vdash_g (c_f,\, 0,\, \perp)$$

  has similar effect with the **destroy event** of an agent α by gray hole $g \in \mathbb{G}$:

  $$(s,\, q_{_{GH}},\, in) \vdash_g^{\alpha} (s_{_{GH}},\, q_{_{GH}},\, 0)$$

  since after the ignore event $t(\alpha)$ will not be involved in any other event in a similar way that agent α after been destroyed will not participate in any other event.

Proposition 3.17. Let $\mathcal{A} = (\vdash_p^{\alpha})_{a \in \mathbb{A},\, p \in \mathbb{P}}$ be a mobile agent algorithm implemented on the mobile agent system with gray holes $(\mathbb{A},\, \mathbb{P},\, \mathbb{G},\, \mathbb{S},\, \pi_0,\, \lambda)$. Let $\mathcal{D} = (\mathcal{D}_p)_{p \in P}$ be the message passing algorithm defined above on the message passing system with omission failures $(P,\, C,\, O,\, \lambda')$. Then algorithm $\mathcal{D}$ of the message passing system simulates algorithm $\mathcal{A}$ of the mobile agent system.

Proof. Similarly with the proof of proposition 3.5 with the additional remarks that:

1. $|O| \leq |\mathbb{G}|$
2. there exists an execution $\mathcal{E}_{\mathcal{A}}$ of $\mathcal{A}$ that the times that a process $p \in O$ crashes and resumes the execution are the same as the times that the corresponding execution place $p \in \mathbb{G}$ behaves as a black hole or a safe execution place.

□

Simulation of a GH by OF

Gray holes can be simulated by omission failures. In this case:

- The **gray hole event** of a gray hole $g \in \mathbb{G}$:

  $$("gray", q_{GH}, in^*) \vdash_g (\sharp, q_{GH}, 0)$$

  is simulated in the message passing model as an **stop event**:

  $$(c, in^*, "stop") \vdash_g (c_o, 0, \perp)$$

  where $c = (q, 0, \sharp)$, $c_o = (q_{GH}, 0, \sharp)$ and in^* is the special port through which the gray hole event arrives from the external environment.

- The **safe node event** of a gray hole $g \in \mathbb{G}$:

  $$("safe", q_{GH}, in^*) \vdash_g (\sharp, q', 0)$$

  is simulated in the message passing model as an **resume event**:

  $$(c_o, in^*, "resume") \vdash_g (c, 0, \perp)$$

  where $c_o = (q_{GH}, 0, \sharp)$, $c = (q', 0, \sharp)$ and in^* is the special port through which the gray hole event arrives from the external environment.

- The **destroy event** of an agent $\alpha \in \mathbb{A}$ by a gray hole $g \in \mathbb{G}$:

  $$(s, q_{GH}, in) \vdash_g^{\alpha} (s_{GH}, q_{GH}, 0)$$

  is simulated in the message passing model as an **ignore event**:

  $$(c_o, in, s) \vdash_g (c_o, 0, \perp)$$

  where $c_o = (q_{GH}, 0, \sharp)$, and in is the port through which token $t(\alpha)$ with state s is received.
  We note that after the destroy event agent α terminates and won't be involved in any other event. In the simulation, token $t(\alpha)$ will arrive at g and will be ignored by the faulty process b and there will be no other event involving token $t(\alpha)$.

The simulation is depicted in figure 3.18

Termination

Lemma 3.18. Let $\mathcal{A} = (\vdash_p^{\alpha})_{a \in \mathbb{A},\, p \in \mathbb{P}}$ be a mobile agent algorithm that tolerates t gray holes implemented $(\mathbb{A}, \mathbb{P}, \mathbb{G}, \mathbb{S}, \pi_0, \lambda)$, where $|\mathbb{G}| \leq t$. Let $\mathcal{D} = (\mathcal{D}_p)_{p \in P}$ be the message passing algorithm defined above on the message passing system with omission failures (P, C, O, λ'), where $|O| \leq t$ and $C = \mathbb{S}$. If algorithm $\mathcal{A}$ has the termination property then algorithm $\mathcal{D}$ has the termination property.

Proof. The proof is similar with the proof of lemma 3.6 □

Complexity

Message Complexity

Proposition 3.19. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm that tolerates t gray holes implemented on the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{G},\, \mathbb{S},\, \pi_0,\, \lambda)$, where $|\mathbb{G}| \leq t$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system with omission failures $(P,\, C,\, O,\, \lambda')$, where $|O| \leq t$ and $C = \mathbb{S}$.

Then the **total number of messages exchanged** during the execution of algorithm $\mathcal{D}$ is:

$$MSG(\mathcal{D}, C) = \sum_{\alpha\in\mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) = TotalMove(\mathcal{A}, \mathbb{S})$$

where $Move(\alpha, \mathcal{A}, \mathbb{S})$ is the move complexity of agent $\alpha \in \mathbb{A}$ in the execution of $\mathcal{A}$. The total number of messages sent to the set of faulty processes O is at most k-1, where $k = |\mathbb{A}|$. The **total size of messages exchanged** is

$$Bit(\mathcal{D}, C) = \sum_{\alpha\in\mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$$

where $LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$ is the local complexity of agent $\alpha \in \mathbb{A}$, i.e. the size of notebook of α.

Proof. If in the simulation algorithm $\mathcal{D}$ there is a send event for process $p \in P \setminus O$ and token $t(\alpha),\ \alpha \in \mathbb{A}$ then in algorithm $\mathcal{A}$ agent $\alpha \in \mathbb{A}$ departs from the execution place $p \in \mathbb{P} \setminus \mathbb{G}$. The size of each message $t(\alpha)$ in $\mathcal{D}$ equals the size of the $notebook_\alpha$ of the corresponding agent α or $state(\alpha)$.

If in the simulation algorithm $\mathcal{D}$ there is a send event to a process with omission failure $g \in O$, then in algorithm $\mathcal{A}$ agent $\alpha \in \mathbb{A}$ arrives to gray hole $g \in \mathbb{G}$ and is either destroyed or is executed normally. □

Time Complexity

Proposition 3.20. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm that tolerates t gray holes implemented on the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{G},\, \mathbb{S},\, \pi_0,\, \lambda)$, where $|\mathbb{G}| \leq t$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system with omission failures $(P,\, C,\, O,\, \lambda')$, where $|O| \leq t$ and $C = \mathbb{S}$. Then the **time complexity** of algorithm $\mathcal{D}$ is:

$$Time(\mathcal{D}, C) = Time(\mathcal{A}, \mathbb{S})$$

where $Time(\mathcal{A}, \mathbb{S})$ is the time complexity of mobile agent algorithm $\mathcal{A}$.

Proof. The proof is similar with the proof of proposition 2.4 □

Space Complexity

Proposition 3.21. Let $\mathcal{A} = (\vdash_p^\alpha)_{a\in\mathbb{A},\, p\in\mathbb{P}}$ be a mobile agent algorithm that tolerates t gray holes implemented on the mobile agent system $(\mathbb{A},\, \mathbb{P},\, \mathbb{G},\, \mathbb{S},\, \pi_0,\, \lambda)$, where $|\mathbb{G}| \leq t$. Let $\mathcal{D} = (\mathcal{D}_p)_{p\in P}$ be the message passing algorithm defined above on the message passing system with omission failures $(P,\, C,\, O,\, \lambda')$, where $|O| \leq t$ and $C = \mathbb{S}$.
Then the **local space complexity** of algorithm $\mathcal{D}$ is:

$$LocMem(\mathcal{D}, C) = \max_{p\in\mathbb{P}} LocMemWB(p, \mathcal{A}, \mathbb{S}) + \max_{\alpha\in\mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$$

where $LocMemWB(p, \mathcal{A}, \mathbb{S})$ is the local space complexity of execution place $p \in \mathbb{P}$ and $LocMemAg$ is the local space complexity of agent $\alpha \in \mathbb{A}$ of algorithm $\mathcal{A}$ on the navigation system $\mathbb{S}$.
The **total space complexity** of algorithm $\mathcal{D}$ is:

$$Mem(\mathcal{D}, C) = \sum_{p \in \mathbb{P}} LocMemWB(p, \mathcal{A}, \mathbb{S}) + \sum_{\alpha \in \mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$$

Proof. The proof is similar with the proof of proposition 2.5 □

The simulation is summarised in tables 3.7, 3.8 and figures 3.17. The simulation of Gray Holes by omission failures is depicted in figure 3.18. The complexity of the simulation is summarised in table 3.9

Table 3.7: Simulation of Mobile Agent Algorithm $\mathcal{A}$ with gray holes by Message Passing Algorithm $\mathcal{D}$ with omission failures

Mobile Agent Model with gray holes	**Message Passing Model with omission failures**
System: $(\mathbb{A}, \mathbb{P}, \mathbb{G}, \mathbb{S}, \pi_0, \lambda)$ $\mathbb{S} = (V, E, \delta)$ navigation subsystem $\mathbb{P}$ or V : execution places, equipped with a whiteboard E : migration ports $\delta_u : N_G(u) \to [1, deg_G(u)] : u \in V$ port labelling function λ : initial states of places and agents $\mathbb{A}$: set of k asynchronous mobile agents. $\pi_0 : \mathbb{A} \to V$ initial placement of agents. $\pi : \mathbb{A} \to V$ location of agents.	**System:** (P, C, O, λ') $C = (V, E, \delta)$ communication subsystem P or V : asynchronous processes E : communication channels $\delta_u : N_G(u) \to [1, deg(u)] : u \in V$ port labelling function λ' : initial states of processes $\lambda'(p) = \begin{cases} (\lambda(p), 1, \lambda(\alpha)), & \text{if } \pi_0(p) = \alpha \\ (\lambda(p), 0, \sharp), & \text{otherwise} \end{cases}, p \in P$ k tokens π_0 : initial placement of tokens. π : location of tokens.
Gray hole $g \in \mathbb{G} \subsetneq \mathbb{P}$**:** a stationary process that chooses whether to act as a black hole or a safe execution place s_{GH}: the state of the destroyed agent	**Faulty process** $g \in O \subsetneq P$**:** a process that might skip to execute some steps of the algorithm $state(g) = s_o = (q_{GH}, 0, \sharp)$
Mobile agent algorithm $\mathcal{A}$ $Q_{\mathbb{A}}$: set of possible states of $\alpha \in \mathbb{A}$ $Q_{\mathbb{P}}$: set of possible states of $p \in \mathbb{P}$ $I_{\mathbb{A}} \subseteq Q_{\mathbb{A}} \setminus \{s_{GH}\}$: set of initial states of α $I_{\mathbb{P}} \subseteq Q_{\mathbb{P}}$: set of initial states of $p \in \mathbb{P}$ $state^{\mathcal{A}}(p),\ p \in \mathbb{P}$ $state^{\mathcal{A}}(\alpha),\ \alpha \in \mathbb{A}$ $\mathbb{M}$: multiset of agents in transit	**Simulation message passing algorithm** $\mathcal{A}$ $\mathcal{M} = Q_{\mathbb{A}} \setminus \{s_{GH}\}$: set of possible messages $Q = Q_{\mathbb{P}} \times \{0,1\} \times Q_{\mathbb{A}} \cup \{\sharp\} \setminus \{s_{GH}\}$: set of possible states of $p \in P$ $I = I_{\mathbb{P}} \times \{0,1\} \times I_{\mathbb{A}} \cup \{\sharp\}$: set of initial states of $p \in P$ $state(p) =$ $= \begin{cases} (state^{\mathcal{A}}(p), 1, state^{\mathcal{A}}(\alpha)), & \pi(p) = \alpha,\ p \in F \\ (state^{\mathcal{A}}(p), 0, \sharp), & \text{otherwise} \end{cases}$ $M = \mathbb{M}$: multiset of messages in transit

Table 3.8: Simulation of Mobile Agent Algorithm $\mathcal{A}$ with gray holes by Message Passing Algorithm $\mathcal{D}$ with Omission failures

Mobile Agent Model with gray holes	**Message Passing Model with omission faults**
Events of the relation $\vdash_p^\alpha$ of $\mathcal{A}$ $h: events(\mathcal{A}) \to events(\mathcal{D})$	**Events of the relation $\vdash_p$ of $\mathcal{D}$**
if $p \in \mathbb{P} \setminus \mathbb{G}$, $\alpha \in \mathbb{A}$:	if $p \in P \setminus O$:
departure event: $(s, q, 0) \vdash_p^\alpha (s', q', out)$	**send event:** $\big((q,1,s), 0, \perp\big) \vdash_p \big((q',0,\sharp), out, s'\big)$
arrival event: $(s, q, in) \vdash_p^\alpha (s', q', 0)$	**receive event:** $\big((q,0,\sharp), in, s\big) \vdash_p \big((q',1,s'), 0, \perp\big)$
α **remains at** p**:** $(s, q, 0) \vdash_p^\alpha (s', q', 0)$	**internal event:** $\big((q,1,s), 0, \perp\big) \vdash_p \big((q',1,s'), 0, \perp\big)$
if $p \in \mathbb{G}$:	if $p \in O$:
gray hole event: $(\sharp, q, in^*) \vdash_p (\sharp, q_{GH}), 0)$	**stop event:** $((q,0,\sharp), in^*, \text{"}crash\text{"}) \vdash_p ((q_{BH},0,\sharp), 0, \perp)$
safe node event: $(\sharp, q_{GH}), in^*) \vdash_p (\sharp, q, 0)$	**resume event:** $((q_{GH}),0,\sharp), in^*, \text{"}resume\text{"}) \vdash_p ((q',0,\sharp), 0, \perp)$
if $p \in \mathbb{G}$ and $state^{\mathcal{A}}(p) = q_{GH}$	if $p \in O$ and $state(p) = c_o = (q_{BH}, 0, \sharp)$
the only applicable event is the **destroy event:** $(s, q_{GH}, in) \vdash_p^\alpha (s_{GH}, q_{GH}, 0)$	and the only applicable event is the **ignore event:** $\big((q_{GH},0,\sharp), in, s\big) \vdash_p \big((q_{GH},0,\sharp), 0, \perp\big)$

Table 3.9: Complexity of the Simulation of the Mobile Agent Algorithm $\mathcal{A}$ by Message Passing Algorithm $\mathcal{D}$

Message Complexity	$MSG(\mathcal{D}, C) = \sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) = TotalMove(\mathcal{A}, \mathbb{S})$
Bit Complexity	$Bit(\mathcal{D}, C) = \sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$
Time Complexity	$Time(\mathcal{D}, C) = Time\big(\mathcal{A}, \mathbb{S}\big)$
Local Space Complexity	$LocMem(\mathcal{D}, C) = \max_{p \in \mathbb{P}}\{LocMemWB(\alpha, \mathcal{A}, \mathbb{S})\} + \max_{\alpha \in \mathbb{A}}\{LocMemAg(\alpha, \mathcal{A}, \mathbb{S})\}$
Total Space Complexity	$Mem(\mathcal{D}, C) = \sum_{p \in \mathbb{P}} LocMemWB(\alpha, \mathcal{A}, \mathbb{S}) + \sum_{\alpha \in \mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$

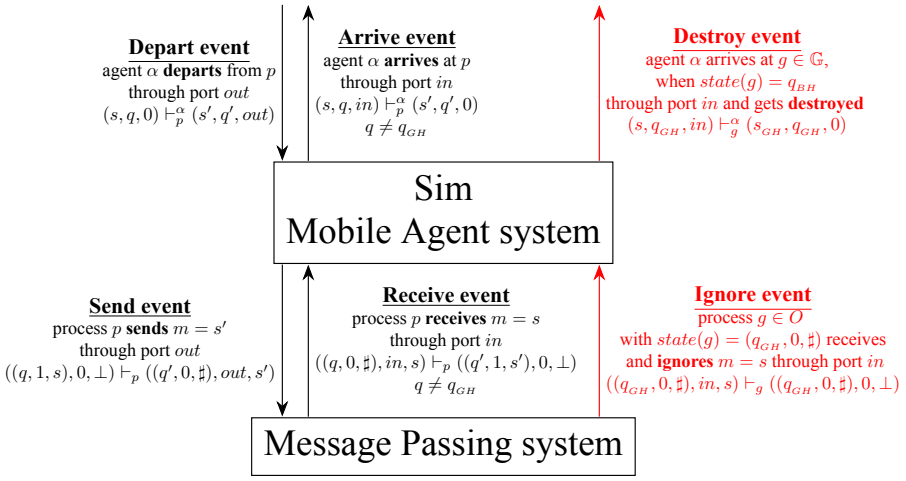

Figure 3.17: Simulation of Mobile agent system with gray holes by a Message Passing system with omission failures, where the inputs of mobile agent system (depart events) are transformed into inputs of the message passing system (send events) and the outputs of the message passing system (receive or ignore events) are transformed into outputs of the mobile agent system (arrive or destroy events). Running the simulation algorithm on top of the message passing system with omission failures produces the same appearance as does running the algorithm on top of the mobile agent system with gray holes.

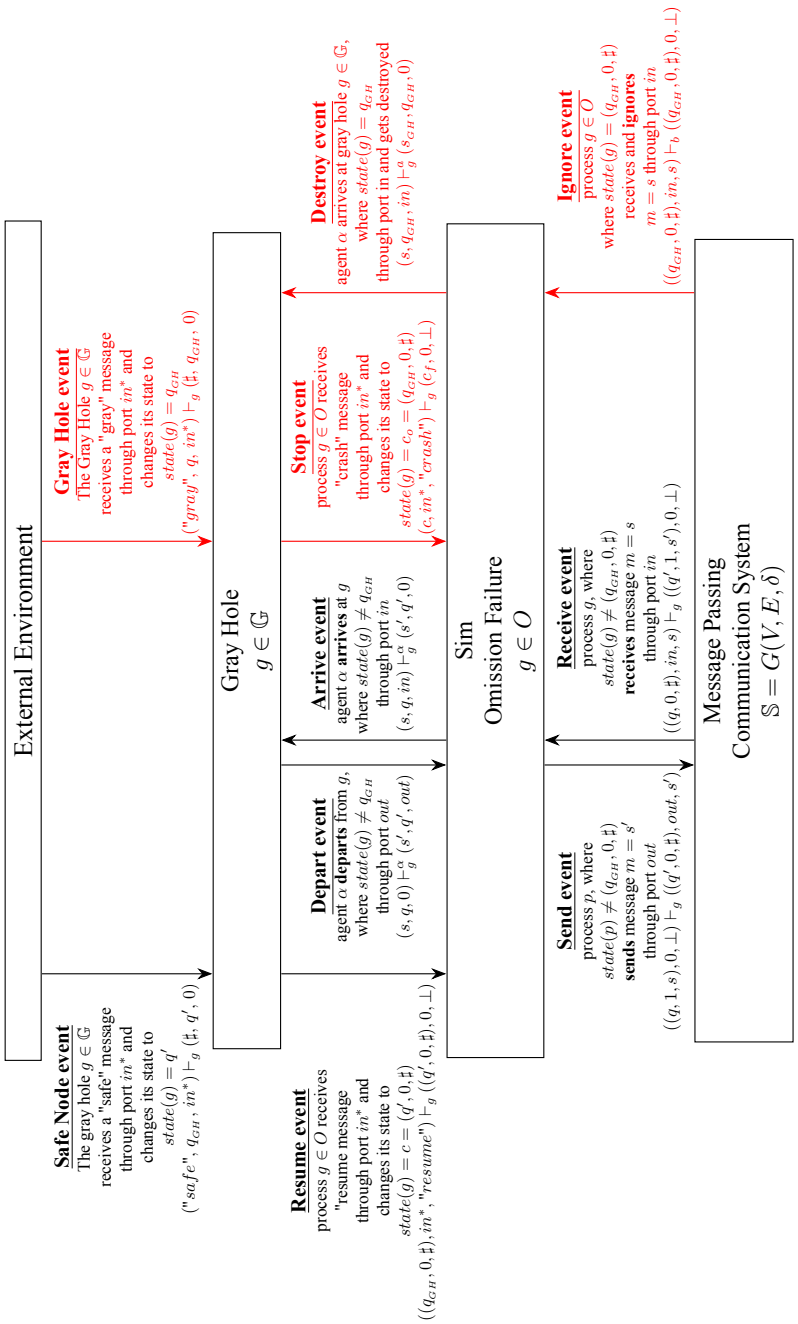

Figure 3.18: Simulation of gray holes by omission failures. The input events of the gray hole, "gray hole event" and "safe node event", from the external environment are transformed to input events of the omission failure, "stop event" and "resume event" respectively. The input events of the omission failure, "ignore event" and "receive event", from the communication system are transformed into "input events" of the gray hole, "destroy event", and "arrive event" respectively and the output event of the gray hole, "depart event" to the omission failure is transformed into an output event of the omission failure, "send event" to the communication system.

3.8 Summarizing tables of the simulations

The simulations described between the mobile agent model and the message passing model and vice versa together with the complexity overhead are presented in the following tables.

3.8.1 Simulation of MP Algorithms by MA systems

Table 3.10: Table of simulations of asynchronous message passing algorithm $\mathcal{D}$ by mobile agent systems. We denote as $|E|$ the total number of edges of G, $|V| = |P|$ the number of processes, $k = |\mathbb{A}|$ the number of agents in the mobile agent system, f the total number of crashed agents and $MSG(\mathcal{D}, C)$ the total number of messages exchanged in the message passing algorithm $\mathcal{D}$

Simulation of Message Passing Algorithms by mobile agents	Agent Complexity $Agents(\mathcal{A}, \mathbb{S})$	Move Complexity $Move(\mathcal{A}, \mathbb{S})$	Local Space Complexity $LocMemWB(\mathcal{A}, \mathbb{S})$	Local Space Complexity $LocMemAg(\mathcal{A}, \mathbb{S})$
MP by MA [9]	$\lvert\mathbb{A}\rvert = k \geq 1$ agents ♯ crashed agents $\leq k-1$	$O\big(\lvert E\rvert + MSG(\mathcal{D}, C) \cdot \lvert V\rvert\big)$	$O(\Delta) + MSG(\mathcal{D}, C) + LocMem(\mathcal{D}, C)$	$O(\lvert V\rvert \log \Delta + \max_{m \in \mathbb{M}} \lvert m\rvert)$
Synchronous MP by MA	"	$O\big(Time(\mathcal{D}, C) \cdot \lvert V\rvert + \lvert E\rvert\big)$	"	"
MP by MA with agent crashes [13]	"	$O\big((\lvert E\rvert + \lvert V\rvert \cdot k) \cdot MSG(\mathcal{D}, C)\big)$	"	"
MP by MA with agent crashes [13] (non-anonymous)	"	$O\big((\lvert E\rvert + \lvert V\rvert \cdot MSG(\mathcal{D}, C)) \cdot k\big)$	"	"
MP by MA with f agent crashes [19] (non-anonymous)	$\lvert\mathbb{A}\rvert = k \geq 1$ agents ♯ crashed agents = $f \leq k-1$	$O\big((\lvert E\rvert + MSG(\mathcal{D}, C)) \cdot f\big)$	"	"
MP with ADP by BH MA	$\lvert\mathbb{A}\rvert \geq k \geq 1$ agents $\mathcal{D}$ sends $\leq k-1$ messages to ADP $\lvert\mathbb{B}\rvert = \lvert D\rvert$	$O\big((\lvert E\rvert + \lvert V\rvert \cdot k) \cdot MSG(\mathcal{D}, C)\big)$	"	"

3.8.2 Simulation of MA Algorithms by MP systems

Table 3.11: Table of simulations of asynchronous mobile agent algorithms $\mathcal{A}$ by message passing systems. We denote as $s(\alpha)$, $\alpha \in \mathbb{A}$, the number of moves of agent $\alpha \in \mathbb{A}$ in mobile agent algorithm $\mathcal{A}$

Simulation of Mobile Agent Algorithms by Message passing Systems	Message Complexity $MSG(\mathcal{D}, C)$	Bit Complexity $Bit(\mathcal{D}, C)$	Time Complexity $Time(\mathcal{D}, C)$	Space Complexity $Mem(\mathcal{D}, C)$
MA by MP [9]	$\sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S})$	$\sum_{\alpha \in \mathbb{A}} \left(Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha, \mathcal{A}, \mathbb{S}) \right)$	$Time(\mathcal{A}, \mathbb{S})$	$\sum_{p \in \mathbb{P}} LocMemWB(p, \mathcal{A}, \mathbb{S}) + \sum_{\alpha \mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$
BH MA by MP with ADP	$\sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S})$	$\sum_{\alpha \in \mathbb{A}} \left(Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha, \mathcal{A}, \mathbb{S}) \right)$	$Time(\mathcal{A}, \mathbb{S})$	$\sum_{p \in \mathbb{P}} LocMemWB(p, \mathcal{A}, \mathbb{S}) + \sum_{\alpha \mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$
B^+H MA by MP with CF	$\sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S})$	$\sum_{\alpha \in \mathbb{A}} \left(Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha, \mathcal{A}, \mathbb{S}) \right)$	$Time(\mathcal{A}, \mathbb{S})$	$\sum_{p \in \mathbb{P}} LocMemWB(p, \mathcal{A}, \mathbb{S}) + \sum_{\alpha \mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$
GH MA by MP with OF	$\sum_{\alpha \in \mathbb{A}} Move(\alpha, \mathcal{A}, \mathbb{S})$	$\sum_{\alpha \in \mathbb{A}} \left(Move(\alpha, \mathcal{A}, \mathbb{S}) \cdot LocMemAg(\alpha, \mathcal{A}, \mathbb{S}) \right)$	$Time(\mathcal{A}, \mathbb{S})$	$\sum_{p \in \mathbb{P}} LocMemWB(p, \mathcal{A}, \mathbb{S}) + \sum_{\alpha \mathbb{A}} LocMemAg(\alpha, \mathcal{A}, \mathbb{S})$

3.9 Discussion

In the simulations presented we could not obtain a simulation relation of a message passing algorithm with crash or omission failures by a mobile agent system with black$^+$ holes or gray holes. The reason is that in order to obtain such a simulation relation the mobile agents need to find a suitable route to traverse the graph and execute the local algorithm. Since the malicious hosts are black$^+$ holes or gray holes, it is an open question whether the mobile agents can find such a route, execute the local message passing algorithm on every execution place and guarantee that at least one agent survives at the end of the execution and all execution places have been executed.

Therefore, it is of great importance to see whether under several assumptions it is possible to obtain these simulations relations. We note that if the number k of mobile agents is strictly larger than the maximum degree Δ of the graph/communication system and there is at most one malicious host (black$^+$ or gray hole), then we could obtain a simulation relation of a message passing algorithm with one crash or omission failures by mobile agent model with one black$^+$ or gray hole respectively, by combining algorithm 6 with the cautious walk technique presented in [14].

Another open question is whether there exists a simulation relation between the Red Hole Mobile agent model [5] and the Byzantine Message Passing Model. While the two systems seem at first to have many common properties, the correspondence of events of the systems is not straight forward as well as the way that mobile agents will move along the graph besides the presence of red holes and execute the message passing algorithm Furthermore, the simulations presented in the previous chapters were focused in the *LOCAL* and *ASYNC* model. It would be of great interest to analyse the simulations of message passing algorithms by mobile agents of sections2.3, 3.1, 3.3.2 in the *CONGEST* model and compare complexity overhead between the *LOCAL*, the *ASYNC* and the *CONGEST* model.

CHAPTER 4

INSTANTIATIONS

4.1 Problem Definitions

The simulations presented in sections 3.3.1 and 3.3.2 can be used to design MP algorithms and MA algorithms by already existing MA algorithm and MP algorithm respectively. In order to do that we have to define a correspondence between the problems of the message passing systems and the mobile agent systems.

A problem specification $\mathcal{P}$ is defined by a set of input states $in(\mathcal{P})$, a set of output states $out(\mathcal{P})$ and a set of allowable sequences $seq(\mathcal{P})$ of input and output states.

Mobile Agent Gathering Problem and Leader Election Problem in the Message Passing System

In the **Mobile Agent Gathering** or simply **gathering** problem, on a mobile agent system $(\mathbb{A},\ \mathbb{P},\ \mathbb{F}, \mathbb{S},\ \pi_0, \lambda),\ \mathbb{F} \in \{\mathbb{B}, \mathbb{B}^+, \mathbb{G}\}$ such that the initial locations of the agents are distinct, $\pi_0 :\ \mathbb{A} \to \mathbb{P}$ is an injection, all the surviving mobile agents should gather at a single execution place $p \in \mathbb{P}$.

In the **Leader Election** problem, on a message passing system $(P, C, T, \lambda'),\ T \in \{D,\ F,\ O\}$ an algorithm $\mathcal{D}$ solves the leader election problem if all terminal states are elected or not elected and in every admissible execution exactly one processor terminates with elected state and all the remaining processors with not-elected state. The conditions that should be satisfied are the **safety condition**, where processes never disagree and the **liveness condition**, where all processes should eventually agree to only one leader.

If algorithm $\mathcal{A}$ solves the gathering problem, then all surviving agents gather at a single execution place and therefore at the end of the simulation algorithm a single process gathers all the tokens and is elected a leader, and all other processes are not leaders.

If algorithm $\mathcal{D}$ solves the leader election problem, then eventually one processes is elected as leader and all other processes are not leaders, therefore in the simulation of $\mathcal{D}$ in the mobile agent system all surviving agents can gather at the execution place that corresponds to the leader process.

Black/Black$^+$/Gray Search Problem and the Locating the Faulty Process Problem

In the **Black/Black$^+$/Gray Search** (BHS/B$^+$HS/GHS) problem, on a mobile agent system $(\mathbb{A},\ \mathbb{P},\ \mathbb{F}, \mathbb{S},\ \pi_0, \lambda),\ \mathbb{F} \in \{\mathbb{B}, \mathbb{B}^+, \mathbb{G}\}$ the mobile agents $\mathbb{A}$, (that can be initially co-located, $\pi_0(\alpha) = p \in \mathbb{P},\ \forall \alpha \in \mathbb{A}$ or distinctly located, $\pi_0 :\ \mathbb{A} \to \mathbb{P}$ is an injection), all surviving agents identify the position of the black/black$^+$/gray/red hole.

In the **Faulty Process Location** problem, on a message passing system (P, C, T, λ'), $T \in \{D,\ F,\ O\}$, all processes identify the location of the faulty process(crash failure, omission failure, byzantine failure).

If algorithm $\mathcal{A}$ solves the Black/Black$^+$/Gray Search problem, then all surviving agents know the location of the malicious host, therefore in the simulation in the message passing system, at least one process has a token that knows the position of the faulty process and by broadcasting it he can notify all the processes about the location of the faulty process.

If Algorithm $\mathcal{D}$ solves the Faulty Process Location problem, then eventually all processes know the location of the faulty process and therefore in the simulation of $\mathcal{D}$ in the mobile agent system all surviving agents know the location of the malicious host.

Periodic Data Retrieval Problem in the mobile agent model with Black/Black$^+$/Gray Holes and Periodic Data Collection Problem in the message passing Model with Always Dead Processes/Crash/Omission Failures

In the **periodic data retrieval** problem [21, 5] we assume that each execution place generates an infinite sequence of data over time. The agents aim in delivering the data from the safe nodes to the homebase. An algorithm $\mathcal{A}$ is correct for the periodic data retrieval problem on the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{F}, \mathbb{S}, \pi_0, \lambda),\ \mathbb{F} \in \{\mathbb{B}, \mathbb{B}^+, \mathbb{G}\}$ if for every execution $\mathcal{E}$ of $\mathcal{A}$ for every execution place $p \in \mathbb{P}$ and for every time t there exists a time $t' > t$ such that the homebase contains all data that where generated by p up to time t.

In the **periodic data collection** problem we assume that each process generates an infinite sequence of data over time. The processes aim in sending the data to some specially marked processes. An algorithm $\mathcal{D}$ is correct for the periodic data retrieval problem on the message passing system $(P, C, T, \lambda'),\ T \in \{D,\ F,\ O\}$ is for every execution $\mathcal{E}$ of $\mathcal{D}$ for every process $p \in P$ and for every time t there exists a time $t' > t$ such that the specially marked processes has received all data that where generated by p up to time t.

If algorithm $\mathcal{A}$ is correct for the the Periodic Data Retrieval problem, then $\forall p \in \mathbb{P},\ t > 0\ \exists\, t' > t$ such that the homebase contains all data generated by p up to time t. Therefore in the simulation of $\mathcal{A}$ in the message passing system $\forall p \in P,\ t > 0\, \exists\, t'' > t$ such that the homebase contains all data generated by p up to time t.

If algorithm $\mathcal{D}$ is correct for the Periodic Data Collection problem, then $\forall p \in P,\ t > 0\, \exists\, t' > t$ such that the homebase contains all data generated by p up to time t. Therefore in the simulation of $\mathcal{D}$ in the mobile agent system $\forall p \in P,\ t > 0\, \exists\, t'' > t$ such that the homebase contains all data generated by p up to time t.

Consensus problem in the Message Passing Model and Coordination problem in the Mobile Agent Model

In the **Consensus** problem in a message passing system $(P, C, T, \lambda'),\ T \in \{D,\ F,\ O\}$ processes should agree on a common course of actions. Each process has an input value

x_i and should produce an output or decision y_i. A solution to the consensus problem should satisfy the following:

- **termination**: For every admissible execution each process $p_i \in P$ outputs a decision value y_i.
- **agreement**: For every execution if y_i, y_j are outputs of processes $p_i, p_j \in P \setminus F$, $i \neq j$ then $y_i = y_j$.
- **validity**: For every execution if $x_i = v$, $\forall p_i \in P$ and process p_i outputs decision value y_i then $y_i = v$.

In the **Coordination** problem in a mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{F}, \mathbb{S}, \pi_0, \lambda)$, $\mathbb{F} \in \{\mathbb{B}, \mathbb{B}^+, \mathbb{G}\}$ mobile agents should agree on a common value and write at the whiteboard of each execution place the agreed value. Each execution place $p_i \in \mathbb{P}$ has an input value x_i and after the termination of the algorithm should produce an output or decision value y_i. A solution to the coordination problem should satisfy the following:

- **termination**: For every admissible execution each execution place $p_i \in \mathbb{P}$ outputs a decision value y_i.
- **agreement**: For every execution if y_i, y_j are outputs of processes p_i, $p_j \in \mathbb{P} \setminus \mathbb{F}$ then $y_i = y_j$.
- **validity**: For every execution of $x_i = v$, $\forall p_i \in \mathbb{P}$ and process p_i outputs a decision value y_i, then $y_i = v$.

If algorithm $\mathcal{D}$ solves consensus in the message passing system, then all non faulty processes agree and decide on a common value and therefore in the simulation in the mobile agent system all execution places have the same value after the termination of the simulation, and the termination, agreement and validity conditions are satisfied.

If algorithm $\mathcal{A}$ solves the coordination problem in the mobile agent system, then all execution places eventually have the same value on their whiteboards, therefore in the simulation in the message passing system all processes agree on a common value and the termination, agreement and validity conditions are satisfied.

k-set Consensus problem in the Message Passing Model and k-set Coordination problem in the Mobile Agent Model

The k-**set Consensus** is a generalization of the consensus and agreement problem and was introduced in [10]. In the k-set Consensus problem in a message passing system (P, C, T, λ'), $T \in \{D, F, O\}$ each process has an input value x_i and should produce an output or decision y_i. A solution to the k-set consensus problem should satisfy the following:

- **termination:** For every admissible execution each process $p_i \in P$ outputs a decision value
- k-**agreement:** For every execution the set of values decided by correct processes has size at most k.
- **validity:** For every execution, the decision value of any correct process must be an input of some correct process.

In the k-**set Coordination** problem in a mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{F}, \mathbb{S}, \pi_0, \lambda), \mathbb{F} \in \{\mathbb{B}, \mathbb{B}^+, \mathbb{G}\}$ each execution place $p_i \in \mathbb{P}$ has an input value x_i and after the termination each execution place should have a decision value y_i. A solution to the k-set coordination problem should satisfy the following:

- **termination:** For every admissible execution each execution place $p_i \in \mathbb{P}$ outputs a decision value y_i.
- k-**agreement:** For every execution the set of values decided by correct processes has size at most k.
- **validity:** For every execution the decision value of any safe execution place must be an input of some safe execution process.

If algorithm $\mathcal{D}$ solves k-set consensus in the message passing system, then all non faulty processes agree and decide on at most k distinct values and therefore in the simulation in the mobile agent system all execution places have at most k distinct values after the termination of the simulation, and the termination, k-agreement and validity conditions are satisfied.

If algorithm $\mathcal{A}$ solves the k-coordination problem in the mobile agent system, then all execution places eventually have at most k distinct values on their whiteboards, therefore in the simulation in the message passing system all processes agree on at most k distinct values and the termination, k-set agreement and validity conditions are satisfied.

Renaming problem

The renaming problem is a special case of coloring problems and was introduced in [3]. The **renaming** problem, in the message passing system (P, C, T, λ'), $T \in \{D,\ F,\ O\}$ processes have unique identifiers from a large domain and each processor $p_i \in P$ should pick a new name y_i from a smaller domain $[1, \ldots, M]$. An algorithm that solves the renaming problem should satisfy **termination**: that each process $p_i \in P$ eventually decides on a $y_i \in [1, \ldots, M]$ and **uniqueness**: that $\forall p_i,\ p_j \in P, i \neq j,\ y_i \neq y_j$.

In the renaming problem in the mobile agent system $(\mathbb{A},\ \mathbb{P}, \mathbb{F}, \mathbb{S}, \pi_0, \lambda), \mathbb{F} \in \{\mathbb{B}, \mathbb{B}^+, \mathbb{G}\}$ execution places have unique identifiers from a large domain and mobile agents should pick new identifiers for each execution place from a smaller domain $[1, \ldots, M]$. An algorithm that solves the renaming problem should satisfy **termination**: that each execution place $p_i \in P$ eventually gets a new name $y_i \in [1, \ldots, M]$ and **uniqueness**: that $\forall p_i,\ p_j \in \mathbb{P}, i \neq j,\ y_i \neq y_j$.

If algorithm $\mathcal{D}$ solves the renaming problem in the message passing system, then every process eventually acquires a new unique name, and therefore in the simulation in the mobile agent system every execution place eventually gets a new unique name from $[1, \ldots, M]$ that satisfies the termination and uniqueness properties.

If algorithm $\mathcal{A}$ solves the renaming problem in the mobile agent system, then every execution place eventually gets a new unique name, and therefore in the simulation in the mobile agent system every process decides on a new name from the smaller domain $[1, \ldots, M]$ that satisfies the termination and uniqueness properties.

Table 4.1: Correspondence between problems in the message passing and in the mobile agent system

Problems in Message Passing	Problems in Mobile Agent
Leader Election	Mobile Agent Gathering
Identify/Locate the faulty processes	Black/Black$^+$/Red hole Search
Periodic Data Collection	Periodic Data Retrieval
Topology discovery Problem	Graph exploration by mobile agents
Consensus	Coordination
k-set Consensus	k-set Coordination
Renaming	Renaming

4.2 Positive Results

In the following sections we will present how to use the simulations presented to acquire positive and impossibility results by already existing results of the literature.

4.2.1 Positive results on ADP and BH model

Lemma 4.1. [14] The Mobile Agent Gathering Problem can be solved in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$, where $|\mathbb{B}| = 1$, $\mathbb{S}$ is an oriented ring, $k \geq 3$ and agents know the number k of agents in the system in time $3n - 6$ and $3k \cdot \sum_{i=1}^{k}(n-i)$ total agent moves.

Corollary 4.2. The Leader Election Problem can be solved in the message passing system with one always dead process (P, C, D, λ'), where C is an oriented ring, $\lambda'(v) = \begin{cases} (\lambda(v),\ 1, \lambda(\alpha)), & \text{if } v \text{ is the homebase of agent } \alpha, \\ (\lambda(v),\ 0,\ \sharp), & \text{otherwise} \end{cases}$, and processes know the number of tokens k, in $3n - 6$ time units and with $3k \cdot \sum_{i=1}^{k}(n-i)$ total messages exchanged.

Proof. By applying lemma 4.1 to the simulation relation presented in section 3.3.1 □

Theorem 4.3. [15] The Mobile Agent Gathering problem of $k-2$ agents can be solved in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$, where $|\mathbb{B}| = 1$, $\mathbb{S}$ is an anonymous oriented ring, agents know $|\mathbb{P}| = n$ and $|\mathbb{A}| = k \geq 4$ in $8(n-2)$ time steps and with $4n^2 + nk - k^2/2 + O(n) + O(k^2)$ agent moves.

Corollary 4.4. The Leader Election Problem can be solved in the message passing system with one always dead process (P, C, D, λ'), where C is an anonymous oriented ring $\lambda'(v) = \begin{cases} (\lambda(v),\ 1, \lambda(\alpha)), & \text{if } v \text{ is the homebase of agent } \alpha, \\ (\lambda(v),\ 0,\ \sharp), & \text{otherwise} \end{cases}$, processes know the number $|P| = n$, k processes have initially token and at most 2 messages are sent to always dead processes in $8(n-2)$ time steps and with $4n^2+nk-k^2/2+O(n)+O(k^2)$ message complexity.

Proof. By applying theorem 4.3 to the simulation relation presented in section 3.3.1. □

Theorem 4.5. [16] The BHS problem can be solved in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$, where $|\mathbb{B}| = 1$ and agents don't know the graph topology, in $O(n^2)$ moves using $\Delta + 1$ agents, where Δ is the maximum degree of the graph.

Corollary 4.6. The problem of locating the faulty process in the message passing system (P, C, D, λ'), where $\lambda'(v) = \begin{cases} (\lambda(v),\ 1, \lambda(\alpha)), & \text{if } v \text{ is the homebase of agent } \alpha, \\ (\lambda(v),\ 0,\ \sharp), & \text{otherwise} \end{cases}$,$|D| = 1$ and processes don't know the graph topology can be solved in with $O(n^2)$ total messages exchanged.

Proof. By applying theorem 4.5 to the simulation presented in section 3.3.1 □

Theorem 4.7. [17] There exists a protocol that solves consensus problem in the asynchronous message passing system (P, C, D, λ'), in which all non faulty processes reach a common decision, the communication system C is a complete graph, provided that the number of always dead processes is $|D| < |P|/2$.

Corollary 4.8. There exists a protocol that solves the coordination problem in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$, where the navigation subsystem $\mathbb{S}$ is complete graph, the number of black holes is $|\mathbb{B}| < |\mathbb{P}|/2$ and the number of mobile agents in the system is $|\mathbb{A}| = |D| \cdot (|P| - |D|) \leq \frac{|\mathbb{P}|^2}{4}$.

Proof. We note that since the graph C of the protocol of theorem 4.7 is complete it has exactly $|D| \cdot (|P| - |D|)$ edges between always dead processes and correct processes. Therefore, we can assume without loss of generality that the protocol of theorem 4.7 sends at most $|D| \cdot (|P| - |D|)$ messages to always dead processes, since by modifying the algorithm of theorem 4.7 such that a process sends a second message to a neighboring process only if it has previously received a message from this process produces the same output. Then running the simulation algorithm 6 on the algorithm of theorem 4.7 solves the problem of coordination in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$ where $\mathbb{S} = C$, $|\mathbb{B}| = |D| < |P|/2$ and $|\mathbb{A}| = |D| \cdot (|P| - |D|) + 1 \leq \frac{|P|^2}{4}$ □

4.2.2 Positive Results on OF and GH

Theorem 4.9. [5] There exists an algorithm that solves the periodic data retrieval problem in an undirected labelled and oriented ring $\mathbb{S}$ in $(\mathbb{A}, \mathbb{P}, \mathbb{G}, \mathbb{S}, \pi_0, \lambda)$, where $|\mathbb{A}| = 4$, $|\mathbb{G}| = 1$.

Corollary 4.10. There exist an algorithm that solves the periodic data collection problem in the message passing model (P, C, O, λ'), $|O| = 1$ that sends at most 3 messages to the crashed process.

Proof. By applying theorem 4.9 to proposition 3.17. □

Table 4.2: Summary of positive results

Lemma 4.1 MA **gathering** problem can be solved in an oriented ring with one BH given that $k = \|\mathbb{A}\| \geq 3$ and agents know k, in time $3n - 6$ and $3k \cdot \sum_{i=1}^{k}(n-i)$ total agent moves	**Corollary**4.2 MP **Leader Election** problem can be solved in an oriented ring with one ADP given that $k \geq 3$ processes have initially a token, in $3n - 6$ time and $3k \cdot \sum_{i=1}^{k}(n-i)$ total messages exchanged
Theorem 4.3 MA **gathering** problem can be solved in an anonymous oriented ring with one BH given that $k = \|\mathbb{A}\| \geq 4$, $k-2$ agents gather, and agents know n, in $8(n-2)$ time steps and $4n^2 + nk - k^2/2 + O(n) + O(k^2)$ agent moves	**Corollary** 4.4 MP **Leader Election** problem can be solved in an anonymous oriented ring with one ADP given that $k \geq 4$ processes have initially token, at most 2 messages are sent to ADP processes know n, in $8(n-2)$ time steps and $4n^2 + nk - k^2/2 + O(n) + O(k^2)$ msgs exchanged
Theorem 4.5 **BHS** problem can be solved in a MA system with one BH when agents don't know the topology, in $O(n^2)$ moves using $\Delta + 1$ agents where $\|\Delta\| = \max_{u \in V} \|N_G(u)\|$	**Corollary**4.6 **ADLP** problem can be solved in a MP system with one ADP when processes don't know the topology with $O(n^2)$ message complexity and $\Delta + 1$ processes have initially a token.
Theorem 4.7 **Consensus** can be solved in asynchronous MP system with less than $n/2$ ADP in complete graphs	**Corollary** 4.8 **Coordination** problem can be solved in asynchronous MA system with less than $n/2$ BH and $\|\mathbb{A}\| = \|D\| \cdot (\|P\| - \|D\|) \leq \frac{\|\mathbb{P}\|^2}{4}$ in complete graphs
Corollary 4.9 $\exists$ Algorithm that solves **periodic data** **retrieval** problem in an undirected labelled and oriented ring with $\|\mathbb{A}\| = k = 4$ agents and one GH	**Corollary** 4.10 $\exists$ algorithm that solves **periodic data collection** in an undirected labelled and oriented ring with one omission faulty process, that sends at most 3 messages to the faulty process.

4.3 Impossibility Results

4.3.1 Impossibility results on ADP and BH model

Simulations are a very powerful tool, not only because they allow us to transfer the positive results of one model to another, but also because they allow us to claim impossibility results.

In sections 3.3.1 and 3.3.2 we proved that the message passing system with always dead processes $(P,\ C,\ D,\ \lambda')$ simulates the mobile agent system with black holes $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$, with respect to the non faulty execution places - processes, where $\lambda'(v) = \begin{cases} (\lambda(v),\ 1, \lambda(\alpha)), \text{ if } v \text{ is the homebase of agent } \alpha, \\ (\lambda(v),\ 0,\ \sharp), \text{ otherwise} \end{cases}$ and that the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$ simulates the message passing system $(P,\ C,\ D,\ \lambda'')$, with respect to the non faulty processes execution places, where $\lambda(v) = \lambda''(v),\ v \in \mathbb{P}$. Therefore we can prove that if there does not exist an algorithm that solves problem Π in the mobile agent system with black holes, then the problem Π cannot be solved in the message passing system with always dead processes, and vice versa.

Proposition 4.11. Let Π be a problem defined on the mobile agent system with black holes $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$. If there does not exist deterministic algorithm to achieve the specifications of Π in $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$, then problem Π cannot be solved in the message passing system with always dead processes $(P,\ C,\ D,\ \lambda')$, where $C = \mathbb{S} =$

$(V,\ E,\ \delta),\ \lambda'(v) = \begin{cases} (\lambda(v),\ 1, \lambda(\alpha)), \text{ if } v \text{ is the homebase of agent } \alpha, \\ (\lambda(v),\ 0,\ \sharp), \text{ otherwise} \end{cases}$ and at most $|\mathbb{A}| - 1$ messages are sent to always dead processes.

Proof. Suppose for contradiction that there exists a message passing algorithm $\mathcal{D}$ that solves Π in $(P,\ C,\ D,\ \lambda')$, where the total number of messages sent to the always dead processes are at most $|\mathbb{A}| - 1$. By algorithm 6, algorithm $\mathcal{D}$ is simulated by algorithm $\mathcal{A}$ in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$. Since algorithm $\mathcal{D}$ achieves problem Π specifications, then algorithm $\mathcal{A}$ achieves problem Π specifications, which yields contradiction. □

Proposition 4.12. Let Π be problem defined on the message passing system with always dead processes $(P,\ C,\ D,\ \lambda')$, where the set of states of the processes is $Q = Q_1 \times \{0,1\} \times Q_2$, where $Q_1,\ Q_2$ are set of states and $\lambda' : P \to I_1 \times \{0,1\} \times I_2, I_1 \subseteq Q_1$, $I_2 \subseteq Q_2$ and $\lambda'(v) = \begin{cases} (q,\ 1, s),\ q \in I_1,\ s \in I_2 \text{ or,} \\ (q,\ 0,\ \sharp),\ q \in I_1,\ \sharp \text{ null} \end{cases}$. If there does not exist deterministic algorithm that achieves the specifications of Π in $(P,\ C,\ D,\ \lambda')$, then problem Π cannot be solved in the mobile agent system with black holes $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$, where $|\mathbb{A}| = \left|\{\lambda'(v) \mid u \in P \wedge \lambda'(v) = (q, 1, s),\ q \in Q_1,\ s \in Q_2\}\right| = k$, $|\mathbb{P}| = |P|$, $|D| = |\mathbb{B}|$, $\mathbb{S} = C = (V,\ E,\ \delta)$, $Q_{\mathbb{P}} = Q_1$, $Q_{\mathbb{A}} = Q_2$, $I_{\mathbb{P}} = I_1$, $I_{\mathbb{A}} = I_2$ and $\lambda(v), v \in \mathbb{P}$ and $\lambda(\alpha),\ \alpha \in \mathbb{A}$ are such that $(\lambda(v), 1, \lambda(\alpha)) = \lambda'(v),\ v \in P$ and $\pi_0(\alpha) = v$, or $(\lambda(v), 0, \sharp) = \lambda'(v),\ v \in P$

Proof. Suppose for contradiction that there exists a mobile agent algorithm $\mathcal{A}$, that solves Π on the system $(\mathbb{A},\ \mathbb{P},\ \mathbb{B},\ \mathbb{S},\ \pi_0, \lambda)$, where $\mathbb{S} = C$ and $\lambda(v) = \lambda(v),\ v \in P$. Then, by the simulation 3.3.1, algorithm $\mathcal{A}$ can be simulated in the message passing system $(P,\ C,\ D,\ \lambda')$. Therefore, the simulation of $\mathcal{A}$ in the message passing system solves Π, that is a contradiction. □

Therefore, by propositions 4.11 and 4.12 we have that the mobile agent system $(\mathbb{A},\ \mathbb{P},\ \mathbb{B},\ \mathbb{S},\ \pi_0,\ \lambda)$ and the message passing system $(P,\ C,\ D,\ \lambda')$ are computationally equivalent.

Instantiations/Examples

The propositions presented in the previous section can be used to acquire and transfer impossibility results from one model to the other. The expression "It is impossible to solve problem Π" means that there does not exist deterministic algorithm $\mathcal{A}$ that correctly terminates for problem Π.

Lemma 4.13. [14, 16] If $|\mathbb{A}| = 1$ the BHS problem cannot be solved.

Corollary 4.14. There does not exist deterministic algorithm that solves the Always Dead Process Location in the message passing system $(P,\ C,\ D,\ \lambda')$, where λ is as described in proposition 4.11, $|D| = 1$ that sends no messages to $d \in D$.

Proof. By applying claim 4.13 to proposition 4.11 □

Lemma 4.15. [14, 16, 12] If $|\mathbb{A}| > 1$, execution places have unique identifiers λ and agents do not know the size of the navigation subsystem, the BHS problem is impossible since it is impossible for the agents to distinguish between a black hole and a slow link or agent.

Corollary 4.16. There does not exist deterministic algorithm that solves the problem of locating an always dead process in the message passing system (P, C, D, λ'),where λ is as described in proposition 4.11, processes have unique identifiers, but there is no knowledge of the size of the communication system.

Proof. By applying claim 4.15 to proposition 4.11 □

Corollary 4.17. [14, 16] It is impossible to verify whether or not there exists a black hole using explicit termination in a ring.

Corollary 4.18. It is impossible to verify whether there exists an always dead process using explicit termination in a ring.

Proof. By applying corollary 4.17 to proposition 4.11 □

Corollary 4.19. [16] If the navigation graph $\mathbb{S}$ has a vertex cut, it is impossible to solve BHS problem.

Corollary 4.20. If the communication graph C has a vertex cut it is impossible to solve the Always Dead Process Location problem.

Proof. By applying corollary 4.19 to proposition 4.11 □

Theorem 4.21. [16] There is a graph $G = (V, E, \lambda)$ with maximum vertex degree $3 \leq \Delta \leq n - 4$ such that without topological information, any algorithm for the Black Hole Search problem in arbitrary networks requires at least $\Delta + 1$ agents in G.

Corollary 4.22. There is a graph $G = (V, E, \lambda)$ with maximum vertex degree $3 \leq \Delta \leq n - 4$ such that without topological information, any algorithm for the always dead process location problem in arbitrary networks sends at least Δ messages in the always dead process in G.

Proof. By applying theorem 4.21 to proposition 4.11 □

Proposition 4.23. [17] There is no deterministic algorithm that solves consensus in the message passing system (P, C, D, λ'), where λ is as described in proposition 4.12, when the number of the always dead processes is $|D| > |P|/2$.

Corollary 4.24. There is no deterministic algorithm that solves coordination problem in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{B}, \mathbb{S}, \pi_0, \lambda)$ when the number of black holes is $|\mathbb{B}| > |\mathbb{P}|/2$.

Proof. By applying proposition 4.23 to proposition 4.12 □

Table 4.3 summarizes some impossibility results that can be obtained by already known results and the propositions 4.11 and 4.12, as a consequence of the simulations presented in sections 3.3.1 and 3.3.2.

Table 4.3: Summary of impossibility results

Black Hole Mobile Agent System	Message Passing with Always Dead
Lemma 4.13 **BHS** problem cannot be solved with only one agent.	**Corollary**4.14 **ADPL** problem cannot be solved without sending any messages to ADP.
Lemma 4.15 **BHS** problem cannot be solved if agents don't know n [1].	**Corollary**4.16 **ADPL** problem cannot be solved if processes don't know n.
Corollary 4.17 It is impossible to verify whether there exists a BH in a ring using explicit termination.	**Corollary** 4.18 It is impossible to verify whether there exists an ADP in a ring using explicit termination.
Corollary 4.19 If the graph has a vertex cut it is impossible to solve **BHS** problem.	**Corollary** 4.20 If the graph has a vertex cut it is impossible to solve **ADPL** problem.
Theorem 4.21 It is impossible to solve **BHS** problem when $\|\mathbb{A}\| \leq \Delta$. [2]	**Corollary** 4.22 It is impossible to solve **ADPL** problem and send at most $\Delta - 1$ messages to BHS. [2]
Corollary 4.24 It is impossible to solve **coordination** problem when $\|\mathbb{B}\| > \|\mathbb{P}\|/2$	**Proposition** 4.23 It is impossible to solve **consensus** when $\|D\| > \|P\|/2$

4.3.2 Impossibility results on CF and B$^+$H model

Proposition 4.25. Let Π be problem defined on the message passing system with crash failures (P, C, F, λ'), where the set of states of the processes is $Q = Q_1 \times \{0,1\} \times Q_2$, where Q_1, Q_2 are set of states and $\lambda' : P \to I_1 \times \{0,1\} \times I_2$, $I_1 \subseteq Q_1$, $I_2 \subseteq Q_2$ and $\lambda'(v) = \begin{cases} (q, 1, s),\ q \in I_1,\ s \in I_2 \text{ or,} \\ (q, 0, \sharp),\ q \in I_1,\ \sharp \text{ is null state} \end{cases}$. If there does not exist deterministic algorithm that achieves the specifications of Π in (P, C, F, λ'), then problem Π cannot be solved in the mobile agent system with black holes $(\mathbb{A}, \mathbb{P}, \mathbb{B}^+, \mathbb{S}, \pi_0, \lambda)$, where $|\mathbb{A}| = \left|\{\lambda'(v) \mid u \in P \wedge \lambda'(v) = (q, 1, s),\ q \in Q_1,\ s \in Q_2\}\right| = k$, $|\mathbb{P}| = |P|$, $|F| = |\mathbb{B}^+|$, $\mathbb{S} = C = (V, E, \delta)$, $Q_{\mathbb{P}} = Q_1$, $Q_{\mathbb{A}} = Q_2$, $I_{\mathbb{P}} = I_1$, $I_{\mathbb{A}} = I_2$ and $\lambda(v), v \in \mathbb{P}$ and $\lambda(\alpha)$, $\alpha \in \mathbb{A}$ are such that $(\lambda(v), 1, \lambda(\alpha)) = \lambda'(v)$, $v \in P$ and $\pi_0(\alpha) = v$, or $(\lambda(v), 0, \sharp) = \lambda'(v)$, $v \in P$

Proof. Suppose for contradiction that there exists a mobile agent algorithm $\mathcal{A}$, that solves Π on the system $(\mathbb{A}, \mathbb{P}, \mathbb{B}^+, \mathbb{S}, \pi_0, \lambda)$, where $\mathbb{S} = C$ and $\lambda(v) = \lambda(v)$, $v \in P$. Then, by the simulation 3.5, algorithm $\mathcal{A}$ can be simulated in the message passing system (P, C, F, λ'). Therefore, the simulation of $\mathcal{A}$ in the message passing system solves Π, that is a contradiction. □

Instantiations/Examples

Theorem 4.26. [17] There is no deterministic algorithm that solves the **consensus** problem in the message passing system (P, C, F, λ') with even one crash failure.

[1] n is the number of vertices of the graph
[2] Δ is the maximum degree of the graph

Corollary 4.27. There is no deterministic algorithm that solves the **coordination** problem in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{B}^+, \mathbb{S}, \pi_0, \lambda)$.

Proof. By applying theorem 4.26 to proposition 4.25. □

Theorem 4.28. [1] If $|F| < n - 1$, the sum of the messages in the failure free runs of any t-resilient protocol for synchronous binary synchronous **consensus/Byzantine Agreement** on the system (P, C, F, λ') under crash failures is at least $n + t - 1$.

Corollary 4.29. There is no t-resilient protocol that solves the synchronous binary **coordination** problem for $|\mathbb{B}^+| < n - 1$, in the system $(\mathbb{A}, \mathbb{P}, \mathbb{B}^+, \mathbb{S}, \pi_0, \lambda)$, that in the failure free runs the total number of agent moves are less than $n + t - 2$.

Proof. By applying theorem 4.28 to proposition 4.25 with the modifications of section 2.2 . □

Theorem 4.30. [20] There is no deterministic algorithm that solves the **renaming** problem in the message passing system (P, C, F, λ'), where $|P| = n+1$, $|F| = f$ the range of input names of processes is $\geq 2n - 1$ and the range of output names of processes is $\leq n + f - 1$.

Corollary 4.31. There is no deterministic algorithm that solves the **renaming** problem in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{B}^+, \mathbb{S}, \pi_0, \lambda)$, where $|\mathbb{P}| = n + 1$, $|\mathbb{B}^+| = f$, the range of input names of execution places is $\geq 2n - 1$ and the range of output names of processes is $\leq n + f - 1$.

Proof. By applying theorem 4.30 to proposition 4.25. □

Proposition 4.32. [20, 7, 30] There is no deterministic algorithm that solves the k**-set consensus** problem in the message passing system (P, C, F, λ'), when $|F| \geq k$.

Corollary 4.33. There is no deterministic algorithm that solves the k**-set coordination** problem in the mobile agent system $(\mathbb{A}, \mathbb{P}, \mathbb{B}^+, \mathbb{S}, \pi_0, \lambda)$, when $|\mathbb{B}^+| \geq k$.

Proof. By applying proposition 4.32 to proposition 4.25. □

Table 4.4: Summary of impossibility results on CF and B$^+$H model

Message Passing with Crash Failures	Black$^+$ Hole Mobile Agent System
Theorem 4.26 $\nexists$ deterministic algorithm for the **consensus** problem with even one crash failure.	**Corollary**4.27 $\nexists$ deterministic algorithm for the **coordination** problem even with one Black$^+$ Hole
Theorem 4.28 The sum of messages in the failure free runs of any t-resilient protocol for synchronous binary **consensus/Byzantine Agreement** under crash failures is at least $n + t - 1$	**Corollary** 4.29 $\nexists$ t-resilient protocol that solves synchronous bunary **coordination** problem that in the failure free runs the total number of agent moves are less than $n + t - 2$
Theorem 4.30 $\nexists$ deterministic algorithm for the **renaming** problem when $\|P\| = n + 1$, $\|F\| = f$ and and the range of input names is $\geq 2n - 1$ the range of output names is $\leq n + f - 1$	**Corollary** 4.31 $\nexists$ deterministic algorithm for the **renaming** problem when $\|\mathbb{P}\| = n + 1$, $\|\mathbb{B}^+\| = f$ and the range of input names is $\geq 2n - 1$ and the range of output names is $\leq n + f - 1$
Proposition 4.32 $\nexists$ deterministic algorithm for the k**-set consensus** when $\|F\| \geq k$	Corollary 4.33 $\nexists$ deterministic algorithm for the k**-set coordination** when $\|\mathbb{B}^+\| \geq k$

4.3.3 Impossibility results on OF and GH model

Proposition 4.34. Let Π be problem defined on the message passing system with crash failures (P, C, O, λ'), where the set of states of the processes is $Q = Q_1 \times \{0,1\} \times Q_2$, where Q_1, Q_2 are set of states and $\lambda' : P \to I_1 \times \{0,1\} \times I_2$, $I_1 \subseteq Q_1$, $I_2 \subseteq Q_2$ and $\lambda'(v) = \begin{cases} (q, 1, s),\ q \in I_1,\ s \in I_2 \text{ or,} \\ (q, 0, \sharp),\ q \in I_1,\ \sharp \text{ is null state} \end{cases}$. If there does not exist deterministic algorithm that achieves the specifications of Π in (P, C, O, λ'), then problem Π cannot be solved in the mobile agent system with black holes $(\mathbb{A}, \mathbb{P}, \mathbb{G}, \mathbb{S}, \pi_0, \lambda)$, where $|\mathbb{A}| = \left|\{\lambda'(v) \mid u \in P \wedge \lambda'(v) = (q, 1, s),\ q \in Q_1,\ s \in Q_2\}\right| = k$, $|\mathbb{P}| = |P|$, $|O| = |\mathbb{G}|$, $\mathbb{S} = C = (V, E, \delta)$, $Q_{\mathbb{P}} = Q_1$, $Q_{\mathbb{A}} = Q_2$, $I_{\mathbb{P}} = I_1$, $I_{\mathbb{A}} = I_2$ and $\lambda(v), v \in \mathbb{P}$ and $\lambda(\alpha)$, $\alpha \in \mathbb{A}$ are such that $(\lambda(v), 1, \lambda(\alpha)) = \lambda'(v)$, $v \in P$ and $\pi_0(\alpha) = v$, or $(\lambda(v), 0, \sharp) = \lambda'(v)$, $v \in P$

Proof. Suppose for contradiction that there exists a mobile agent algorithm $\mathcal{A}$, that solves Π on the system $(\mathbb{A}, \mathbb{P}, \mathbb{G}, \mathbb{S}, \pi_0, \lambda)$, where $\mathbb{S} = C$ and $\lambda(v) = \lambda(v)$, $v \in P$. Then, by the simulation 3.7, algorithm $\mathcal{A}$ can be simulated in the message passing system (P, C, O, λ'). Therefore, the simulation of $\mathcal{A}$ in the message passing system solves Π, that is a contradiction. □

4.4 Conclusion

In this thesis we studied the simulation relations between the message passing model and the mobile agent model, under various assumptions about the types of faults of the two models. The simulations presented are a useful tool to obtain positive and impossibility results from one model to the other and to design algorithms for one model using an already existing algorithm of the other model. Simulations can also be used when designing algorithms that run into complex models, while they are designed into simpler models.

The simulations presented in this thesis give us a tool to do trade offs based on the number of available resources(processes) and the ability of mobility of the resources and in designing algorithms. For example, in the case of testing the performance of a distributed algorithm, a set of k mobile agents can simulate a set of n ($k < n$) stationary agents and vice versa, which can be very helpful in algorithmic design.

Lastly, the simulations of the adversary and its capabilities, that were presented can be used in studying and analysing adversarial strategy and testing the performance of an algorithm according to any adversarial strategy.

BIBLIOGRAPHY

[1] Eugene S. Amdur, Samuel M. Weber, and Vassos Hadzilacos. ``On the Message Complexity of Binary Byzantine Agreement under Crash Failures''. In: *Distributed Comput.* 5.4 (1992), pp. 175–186. DOI: 10.1007/BF02277665. URL: https://doi.org/10.1007/BF02277665.

[2] Hagit Attiya, Amotz Bar-Noy, and Danny Dolev. ``Sharing Memory Robustly in Message-Passing Systems''. In: *J. ACM* 42.1 (1995), pp. 124–142. ISSN: 0004-5411. DOI: 10.1145/200836.200869. URL: https://doi.org/10.1145/200836.200869.

[3] Hagit Attiya, Amotz Bar-Noy, Danny Dolev, Daphne Koller, David Peleg, and Radiger Reischuk. ``Achievable cases in an asynchronous environment''. In: *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*. 1987, pp. 337–346. DOI: 10.1109/SFCS.1987.5.

[4] Hagit Attiya and Jennifer L. Welch. *Distributed computing - fundamentals, simulations, and advanced topics (2. ed.)* Wiley series on parallel and distributed computing. Wiley, 2004. ISBN: 978-0-471-45324-6.

[5] Evangelos Bampas, Nikos Leonardos, Euripides Markou, Aris Pagourtzis, and Matoula Petrolia. ``Improved periodic data retrieval in asynchronous rings with a faulty host''. In: *Theor. Comput. Sci.* 608 (2015), pp. 231–254. DOI: 10.1016/j.tcs.2015.09.019. URL: https://doi.org/10.1016/j.tcs.2015.09.019.

[6] Lali Barrière, Paola Flocchini, Pierre Fraigniaud, and Nicola Santoro. ``Can we elect if we cannot compare?'' In: *SPAA 2003: Proceedings of the Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, June 7-9, 2003, San Diego, California, USA (part of FCRC 2003)*. ACM, 2003, pp. 324–332. DOI: 10.1145/777412.777469.

[7] Elizabeth Borowsky and Eli Gafni. ``Generalized FLP Impossibility Result for T-Resilient Asynchronous Computations''. In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*. STOC '93. San Diego, California, USA: Association for Computing Machinery, 1993, pp. 91–100. ISBN: 0897915917. DOI: 10.1145/167088.167119. URL: https://doi.org/10.1145/167088.167119.

[8] Elizabeth Borowsky, Eli Gafni, Nancy Lynch, and Sergio Rajsbaum. ``The BG Distributed Simulation Algorithm''. In: *Distributed Computing* 14 (July 2001), pp. 127–146. DOI: 10.1007/PL00008933.

[9] Jérémie Chalopin, Emmanuel Godard, Yves Métivier, and Rodrigue Ossamy. ``Mobile Agent Algorithms Versus Message Passing Algorithms''. In: *Principles of Distributed Systems, 10th International Conference, OPODIS 2006, Bordeaux, France, December 12-15, 2006, Proceedings*. Vol. 4305. Lecture Notes in Computer Science. Springer, 2006, pp. 187–201. DOI: 10.1007/11945529_14.

[10] Soma Chaudhuri. ``Agreement is harder than consensus: Set Consensus problems in totally Asynchronous Systems''. In: *Proceedings of the 9th Annual ACM Symposium on Principles of distributed Computing*. 1990, pp. 311–324.

[11] Sandro Coretti, Juan Garay, Martin Hirt, and Vassilis Zikas. ``Constant-Round Asynchronous Multi-Party Computation Based on One-Way Functions''. In: *Proceedings, Part II, of the 22nd International Conference on Advances in Cryptology --- ASIACRYPT 2016 - Volume 10032*. Berlin, Heidelberg: Springer-Verlag, 2016, pp. 998–1021. ISBN: 9783662538890. DOI: 10.1007/978-3-662-53890-6_33. URL: https://doi.org/10.1007/978-3-662-53890-6_33.

[12] Jurek Czyzowicz, Dariusz R. Kowalski, Euripides Markou, and Andrzej Pelc. ``Complexity of Searching for a Black Hole''. In: *Fundam. Informaticae* 71 (2006), pp. 229–242.

[13] Shantanu Das, Paola Flocchini, Nicola Santoro, and Masafumi Yamashita. ``Fault-Tolerant Simulation of Message-Passing Algorithms by Mobile Agents''. In: *Structural Information and Communication Complexity, 14th International Colloquium, SIROCCO 2007, Castiglioncello, Italy, June 5-8, 2007, Proceedings*. Vol. 4474. Lecture Notes in Computer Science. Springer, 2007, pp. 289–303. DOI: 10.1007/978-3-540-72951-8_23.

[14] Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. ``Mobile Search for a Black Hole in an Anonymous Ring''. In: *DISC*. 2001.

[15] Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. ``Multiple Agents RendezVous in a Ring in Spite of a Black Hole''. In: *Principles of Distributed Systems*. Ed. by Marina Papatriantafilou and Philippe Hunel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 34–46. ISBN: 978-3-540-27860-3.

[16] Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. ``Searching for a black hole in arbitrary networks: optimal mobile agents protocols''. In: *Distributed Computing* 19 (2006), pp. 1–99999.

[17] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. ``Impossibility of Distributed Consensus with One Faulty Process''. In: *J. ACM* 32.2 (1985), pp. 374–382. ISSN: 0004-5411. DOI: 10.1145/3149.214121. URL: https://doi.org/10.1145/3149.214121.

[18] Juan A. Garay and Yoram Moses. ``Fully polynomial Byzantine agreement in t + 1 rounds''. In: *Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing* (1993).

[19] Tsuyoshi Gotoh, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. ``How to Simulate Message-Passing Algorithms in Mobile Agent Systems with Faults''. In: *Stabilization, Safety, and Security of Distributed Systems - 19th International Symposium, SSS 2017, Boston, MA, USA, November 5-8, 2017, Proceedings*. Vol. 10616. Lecture Notes in Computer Science. Springer, 2017, pp. 234–249. DOI: 10.1007/978-3-319-69084-1_16.

[20] Maurice Herlihy and Nir Shavit. ``The Asynchronous Computability Theorem for t-Resilient Tasks''. In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*. STOC '93. San Diego, California, USA: Association for Computing Machinery, 1993, pp. 111–120. ISBN: 0897915917. DOI: 10.1145/167088.167125. URL: https://doi.org/10.1145/167088.167125.

[21] Rastislav Královič and Stanislav Miklík. ``Periodic Data Retrieval Problem in Rings Containing a Malicious Host''. In: *Structural Information and Communication Complexity*. Ed. by Boaz Patt-Shamir and Tınaz Ekim. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 157–167. ISBN: 978-3-642-13284-1.

[22] Ajay D. Kshemkalyani and Mukesh Singhal. *Distributed Computing: Principles, Algorithms, and Systems*. 1st ed. USA: Cambridge University Press, 2008. ISBN: 0521876346.

[23] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996. ISBN: 1-55860-348-4.

[24] Nancy A. Lynch and Mark R. Tuttle. ``Hierarchical Correctness Proofs for Distributed Algorithms''. In: *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*. PODC '87. Vancouver, British Columbia, Canada: Association for Computing Machinery, 1987, pp. 137–151. ISBN: 089791239X. DOI: 10.1145/41840.41852. URL: https://doi.org/10.1145/41840.41852.

[25] Euripides Markou and Wei Shi. ``Dangerous Graphs''. In: *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*. Ed. by Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Cham: Springer International Publishing, 2019, pp. 455–515. ISBN: 978-3-030-11072-7. DOI: 10.1007/978-3-030-11072-7_18. URL: https://doi.org/10.1007/978-3-030-11072-7_18.

[26] Gil Neiger and Sam Toueg. ``Automatically increasing the fault-tolerance of distributed algorithms''. In: *Journal of Algorithms* 11.3 (1990), pp. 374–419. ISSN: 0196-6774. DOI: https://doi.org/10.1016/0196-6774(90)90019-B. URL: https://www.sciencedirect.com/science/article/pii/019667749090019B.

[27] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000. ISBN: 0898714648. DOI: 10.1137/1.9780898719772. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898719772.

[28] Kenneth J. Perry and Sam Toueg. ``Distributed agreement in the presence of processor and communication faults''. In: *IEEE Transactions on Software Engineering* SE-12.3 (1986), pp. 477–482. DOI: 10.1109/TSE.1986.6312888.

[29] Michel Raynal. *Fault-Tolerant Message-Passing Distributed Systems: An Algorithmic Approach*. Jan. 2018. ISBN: 978-3-319-94140-0. DOI: 10.1007/978-3-319-94141-7.

[30] Michael Saks and Fotios Zaharoglou. ``Wait-Free k-Set Agreement is Impossible: The Topology of Public Knowledge". In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*. STOC '93. San Diego, California, USA: Association for Computing Machinery, 1993, pp. 101–110. ISBN: 0897915917. DOI: 10.1145/167088.167122. URL: https://doi.org/10.1145/167088.167122.

[31] Tomoko Suzuki, Taisuke Izumi, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. ``Move-optimal gossiping among mobile agents". In: *Theor. Comput. Sci.* 393.1-3 (2008), pp. 90–101. DOI: 10.1016/j.tcs.2007.11.007. URL: https://doi.org/10.1016/j.tcs.2007.11.007.

[32] Gerard Tel. *Introduction to Distributed Algorithms*. 2nd. USA: Cambridge University Press, 2001. ISBN: 0521794838.